Been There, Done That: What Your Mobility Traces Reveal about Your Behavior

Nokia Mobile Data Challenge - Next Place Prediction*

Vincent Etter
LCA 3-4, EPFL
Lausanne, Switzerland

Mohamed Kafsi
LCA 3-4, EPFL
Lausanne, Switzerland

Ehsan Kazemi
LCA 4, EPFL
Lausanne, Switzerland

1. INTRODUCTION

Mobility is a central aspect of our life; the locations we visit reflect our tastes and lifestyle and shape our social relationships. The ability to foresee the places a user will visit is therefore beneficial to numerous applications, ranging from forecasting the dynamics of crowds to improving the relevance of location-based recommendations. To solve the Next Place Prediction task of the Nokia Mobile Data Challenge, we developed several mobility predictors, based on graphical models, neural networks, and decision trees, and explain some of the challenges that we faced. Then, we combine these predictors using different blending strategies, which improve the prediction accuracy over any individual predictor.

The paper is organized as follows: In Section 2, we introduce a framework where we define the notations, the learning process and the prediction performance measure. Then, we briefly present, in Section 3, various models we use to build predictors, and show their individual performance. Finally in Section 4, we describe different blending strategies we implemented.

2. PLACE PREDICTION FRAMEWORK

In the Nokia Mobile Data Challenge, our task is to propose a user-specific predictor that learns from a user's mobility history and predicts, based on the current context, the next location he will visit. The dataset we use consists of data collected from the mobile phones of 80 users, over periods of time varying from a few weeks to two years. A detailed description of the dataset and the collection campaign is available in [4].

We summarize here only some salient characteristics of the data that we believe are critical to the prediction task:

*This material was prepared for the Mobile Data Challenge 2012 (by Nokia) Workshop; June 18-19, 2012; Newcastle, UK.

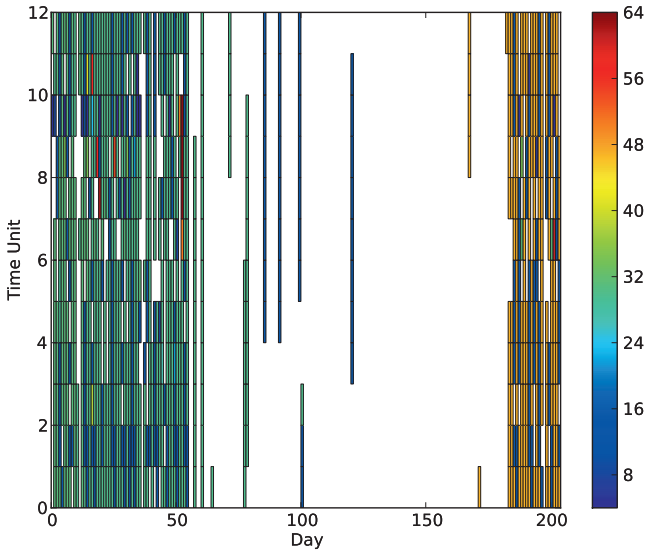

Figure 1: We represent a user's habits as a matrix where each row represents an interval of 2 hours, and each column represents 1 day. Each location is associated with a color. For this user, we observe both a data gap of 60 days and non-stationarity. Note the home change at the end of the observation period.

User Specificity. Although the dataset derives from 80 different users, the data and the prediction task are explicitly user-specific. It is therefore not possible to build joint models over the user population, *i.e.*, to learn from one user to make a prediction for another. For this reason, we build user-specific predictors, and consider each user independently.

Non-Stationarity. We observe a change in users' habits over time. The fact that some users change their home or work location right at the end of the observation period complicates the prediction task.

Data Gaps. We experience, for some users, long periods (up to a few months) with no information about their behaviour. Moreover, as shown in Figure 1, these gaps are sometimes followed by change of mobility habits.

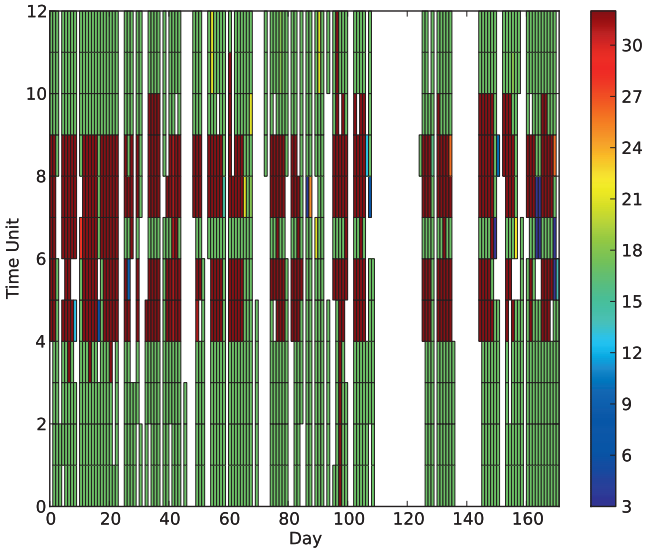

Figure 2: We represent a user's habits as a matrix where each row represents an interval of 2 hours and each column 1 day. Each location is associated with a color. The user's mobility patterns are regular with no change of habits during the observation period (as opposed to the user represented in Figure 1). As expected, the prediction accuracy for this user is high (more than 90%).

Sparsity. The period of observation for some users is too short (less than 15 days) to reflect faithfully the user's mobility patterns.

We believe that taking the above characteristics into account, when designing predictors, has a significant effect on their prediction accuracy.

2.1 Notation

Before formally introducing our predictors, we need to define the variables that describe a user's mobility. During the study period, the user makes N visits of variable duration to L distinct locations, represented by the set $\mathcal{L} = \{1, \ldots, L\}$.

In Table 1, we describe the variables relative to the n^{th} visit. The binary variable $\mathbb{I}_{trust}(n)$ indicates whether the transition from location $X(n)$ to location $X(n+1)$ is trusted, *i.e.*, there is no visit to an intermediary location when moving from $X(n)$ to $X(n+1)$. Discretized starting hour $H_s^k(n)$ and ending hour $H_e^k(n)$ are both computed from a UNIX timestamp t as follows:

$$quantize(t,k) = \left\lfloor \left(\frac{t}{3600} \mod 24 \right) \frac{k}{24} \right\rfloor + 1.$$

The above function depends on a parameter k, which allows us to consider a coarser separation of the day: instead of splitting a day in 24 hours, for example, we can choose to split it in k time periods. For instance, if $k = 2$, $H_s^k(n) \in \{1, 2\}$, with $H_s^k(n) = 1$ corresponding to a visit starting between midnight and noon.

2.2 Learning and Performance Measure

Learning Procedure. For all users, we separate the data into three parts, as illustrated in Figure 3: we define the first 80% of the data as *set A*, the following 10% as *set B* and the last 10% as *set C*. Finally, we call *set D* the undisclosed part of the data over which our predictors will be evaluated. The reason we divide deterministically the dataset is based on the non-stationarity of the user behavior. In fact, we expect *set D* to be much more similar to the end of the dataset than to its beginning. By training our predictors on "past" data and evaluating them on very recent data, we can test whether they are able to adapt to users' change of habits.

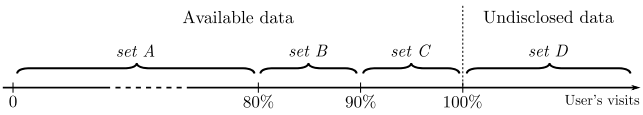

Figure 3: Separation of the user's dataset. We define the first 80% of the user's visits as *set A*, the following 10% as *set B*, and the last 10% as *set C*. Finally, we call *set D* the undisclosed part of the dataset, on which the final performances are computed.

For each predictor, the training is performed in three parts: first, we train over *set A*, and evaluate the performance on *set B*, to compare individual predictors. Then, we train on both *set A* and *set B* and test the prediction accuracy on *set C*, with different blending strategies. Finally, we train on *sets A, B* and *C* in order to predict the samples in *set D*.

Evaluation Metric. To evaluate the performance of a predictor over a set of visits, we consider its prediction accuracy, *i.e.*, the proportion of samples for which it successfully predicts the next location.
First, consider a predictor ϕ: It takes as input v_n, the data corresponding to the n^{th} visit, and outputs a probability distribution P_n^ϕ over the possible next locations. The predicted location $\hat{X}_n^\phi$ is thus defined as

$$\hat{X}_n^\phi = \arg\max_{l \in \mathcal{L}} P_n^\phi(l).$$

We could directly define the output of a predictor as the predicted next location. However, keeping as output a distribution over places allows us to combine predictors in the blending phase, as explained in Section 4.
Finally, we define the prediction accuracy $A_S(\phi)$ of the predictor ϕ over the samples in *set S* as:

$$A_S(\phi) = \frac{1}{|S|} \sum_{i \in S} \mathbb{I}_{\left\{\hat{X}_i^\phi = X(i+1)\right\}},$$

where $|S|$ is the number of samples in *set S*, $X(i+1)$ is the true next location corresponding to the ith visit, and $\mathbb{I}_{\left\{\hat{X}_i^\phi = X(i+1)\right\}}$ is equal to 1 if ϕ correctly predicts the next location, and 0 otherwise.

Definition	Domain	Explanation
$X(n)$	$\mathcal{L}$	Location
$T_s(n)$	$\mathbb{Z}$	Starting time (UNIX timestamp)
$H_s^k(n) = \text{quantize}(T_s(n), k)$	$\{1, \dots, k\}$	Discretized starting hour
$D_s(n) = \text{day}(T_s(n))$	$\{1, \dots, 7\}$	Starting day
$W_s(n) = \text{weekday}(T_s(n))$	$\{0, 1\}$	Indicates whether the visit starts on a weekday
$T_e(n)$	$\mathbb{Z}$	Ending time (UNIX timestamp)
$H_e^k(n) = \text{quantize}(T_e(n), k)$	$\{1, \dots, k\}$	Discretized ending hour
$D_e(n) = \text{day}(T_e(n))$	$\{1, \dots, 7\}$	Ending day
$W_e(n) = \text{weekday}(T_e(n))$	$\{0, 1\}$	Indicates whether the visit ends on a weekday
$\mathbb{I}_{trust}(n)$	$\{0, 1\}$	Indicates whether the transition between visits n and $n+1$ is trusted

Table 1: List of the definitions and domains of the different variables describing the n^{th} visit of a user.

3. PREDICTORS

In this section, we present the techniques we use to build predictors. We briefly describe each method, and summarize their performance in Section 3.4.

3.1 Dynamical Bayesian Network

We introduce a Dynamical Bayesian Network (DBN) to model the mobility patterns of individuals. The rationale behind the model is as follows: the next visit of a user depends on his current location, but also on the starting time of the next visit. The current location is informative especially when the next visit is close in time. However, as the time gap between the visits gets larger, the starting time of the next visit bears increasing importance. As we do not know the starting time of the next visit, the main challenge is to express its randomness, given carefully chosen information about the current visit. As shown in Figure 4, our DBN

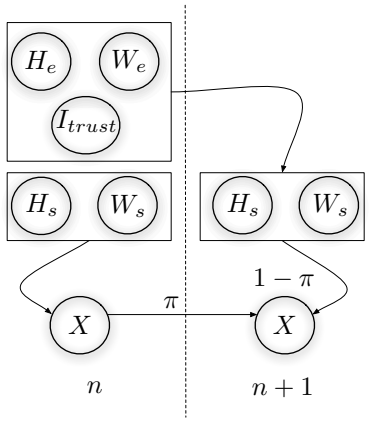

Figure 4: Diagram of a DBN where nodes represent random variables and edges probabilistic dependencies between them.

captures those intuitions: the conditional distribution of the next location $p\left(X(n+1)|X(n), H_e(n), \mathbb{I}_{trust}(n), W_e(n)\right)$ is a mixture of location- and time-dependent distributions

$$\pi p\left(X(n+1)|X(n)\right) + (1-\pi)p\left(X(n+1)|H_e(n), W_e(n), \mathbb{I}_{trust}(n)\right),$$

where $\pi \in [0, 1]$ is the parameter that governs the contribution of each distribution. For ease of notation, we omit the parameter k for the time discretization and assume that it is fixed. The location-dependent component $p(X(n+1)|X(n))$ is simply a first order Markov chain which encodes the frequency of transitions between locations. Using Bayes' rule, we express the time-dependent distribution $p\left(X(n+1)|H_e(n), W_e(n), \mathbb{I}_{trust}(n)\right)$ as

$$\sum_{W_s}\sum_{H_s} p\left(X(n+1)|H_s(n+1), W_s(n+1), H_e(n), W_e(n), \mathbb{I}_{trust}(n)\right) p\left(H_s(n+1), W_s(n+1)|H_e(n), W_e(n), \mathbb{I}_{trust}(n)\right).$$

Note that the conditional distribution

$$p\left(H_s(n+1), W_s(n+1)|H_e(n), W_e(n), \mathbb{I}_{trust}(n)\right)$$

captures the randomness of the starting time of the next visit given the ending time of the current one and the trustiness of the transition. Empirically, we observe that trusted transitions usually imply a shorter interval of time between the visits. By assuming that $X(n+1)$ is independent of $H_e(n)$, $W_e(n)$ and $\mathbb{I}_{trust}(n)$ given $H_s(n+1)$ and $W_s(n)$, we can write

$$\sum_{W_s}\sum_{H_s} p\left(X(n+1)|H_s(n+1), W_s(n+1)\right) p\left(H_s(n+1), W_s(n+1)|H_e(n), W_e(n), \mathbb{I}_{trust}(n)\right).$$

The assumption of independence makes sense, as knowing the time $(H_e(n), W_e(n))$ at which a user leaves the current location is not informative (with respect to the next location $X(n+1)$), given that we know the starting time of the next visit $(H_s(n+1), W_s(n+1))$.

The choice of the model structure and variables is driven by intuition and confirmed by empirical evidence. We tested several variants of our model: For example, we tried to incorporate in our DBN the distribution $p(X(n+1)|X(n), \mathbb{I}_{trust}(n))$ instead of the distribution $p(X(n+1)|X(n))$ but the prediction error increased. Moreover, data sparsity prohibits us from learning more sophisticated distributions.

To use our model for prediction, we estimate the model parameters $p\left(H_s(n+1), W_s(n+1)|H_e(n), W_e(n), \mathbb{I}_{trust}(n)\right)$, $p\left(X(n+1)|H_s(n+1), W_s(n+1)\right)$, $p(X(n+1)|X(n))$, and π. We take two approaches. The first approach is to learn the distributions by counting the frequency of given realisations and then choosing the parameter π that minimizes the prediction error on the test set. The second approach is to formulate the mixture of distributions with respect to a

latent variable $\boldsymbol{z}$: we introduce a N dimensional binary random variable $\boldsymbol{z}$ that indicates, for each visit, the distribution from which it was sampled. In other words, $\boldsymbol{z}_i = 1$ means that the i^{th} visit is sampled from the location dependent distribution, whereas $\boldsymbol{z}_i = 0$ implies that it is sampled from the time-dependent distribution. We then use an *Expectation-Maximization* algorithm [1] to maximize the likelihood of the data with respect to the model parameters.

Aging to Overcome non-Stationarity. In order to reduce the negative impact of non-stationarity on the prediction performance, we introduce an *aging* mechanism, governed by the aging parameter $\lambda \in [0, 1]$. The aging parameter is a multiplicative factor that intervenes in the learning process to reduce the contribution of old samples. We have also implemented an algorithm that detects changes in home location and adapts the learning process accordingly.

3.2 Artificial Neural Networks

We can also consider next place prediction as a classification task: given the current place, and potentially some additional features, we predict the next location. With this approach, we train for each user a 2-layer Artificial Neural Network (ANN) that has N_{in} inputs and N_{hu} hidden units and that outputs a probability distribution over places. Such a network is illustrated in Figure 5.

As input, we encode places as categorical data: we represent each location l as a vector $\boldsymbol{v} \in \{0, 1\}^L$, where $\boldsymbol{v}_i = 1$ if $l = i$, and 0 otherwise. Other attributes, such as $D_e(n)$ or $H_e^k(n)$, can also be included as additional features, and are encoded in a similar way if needed. For example, if we want to use $(X(n), H_s^k(n), D_s(n))$ as inputs, we simply represent them as explained above and concatenate them. The resulting input vector $\boldsymbol{x}$ is thus of size $N_{in} = L + k + 7$. For each user, we consider different subsets of the features described in Table 1, as well as different values for k.

To obtain a probability distribution $\boldsymbol{y} \in [0, 1]^L$ from the output $\boldsymbol{z} \in \mathbb{R}^L$ of the second layer, we use a soft-max transfer function:

$$\boldsymbol{y}_i = \text{softMax}(\boldsymbol{z}_i) = \frac{\exp(\boldsymbol{z}_i)}{\sum_{j=1}^{L} \exp(\boldsymbol{z}_j)}, i \in \{1, \ldots, L\}.$$

A natural loss function to train such a network is the *negative log-likelihood*. For the output $\boldsymbol{y} \in [0, 1]^L$, corresponding to some input $\boldsymbol{x} \in \mathbb{R}^{N_{in}}$ and the ground truth $\boldsymbol{t} \in \{0, 1\}^L$ (where $\boldsymbol{t}_i = 1$ if i is the true next location, and 0 otherwise), we define the loss as:

$$L(\boldsymbol{y}, \boldsymbol{t}) = -\sum_{l=1}^{L} \boldsymbol{t}_l \log(\boldsymbol{y}_l).$$

To find the optimal parameters of the ANN, we minimize the above loss function over the training set.

Implementation. We implement our ANNs by using Torch 5 [2], a machine learning framework written in Lua. We use a stochastic gradient descent [5] to train each ANN, and we use early stopping as a regularization technique. For all users, we empirically found that $N_{hu} = 50$ hidden units were sufficient. To speed up the training, we use hardTanh as the non-linear transfer function between the two layers. It is an approximation of the hyperbolic tangent, that is much faster to evaluate.

Method	Accuracy
Most visited location	35.00%
First order Markov chain	44.00%
DBN	60.07%
ANN	60.83%
GBDT	57.63%

Table 2: Accuracy over *set C* of the different families of predictors, trained on *sets A* and *B*, averaged over all users. For each user, we chose the predictor that yields the best prediction accuracy. Note that this may overestimate the performance of each family in general, as we take the best predictors for *set C* in particular.

3.3 Gradient Boosted Decision Trees

Boosting is a method for combining weak base classifiers in order to build a classifier whose performance is significantly better than the base classifiers. In boosting methods, the base classifiers are trained in sequence, and each of them is trained using a weighted form of the data set, in which the weighing coefficient of each data sample depends on the performance of the previous classifiers. A Gradient Boosted Decision Tree (GBDT) [3] can be used for classification; it is accurate, fast and insensitive to noisy and incomplete data. GBDT is an ensemble of weak decision trees, where all trees consist of two to eight nodes and are learned using boosting methods. The final prediction model is built by adding up contributions of all the individual small trees. Indeed, the model is an ensemble of the trees that becomes more accurate as the number of trees increases.

3.4 Results

We show in Table 2 the prediction accuracy on *set C* for each of the three families of predictors presented above, averaged over all users. For comparison, we also include two baseline predictors: the first always predicts the most visited location, while the second one uses a first order Markov chain. We observe that the three families of predictors have similar performances.

We obtain these results by using only simple features of the visits. In an attempt to improve the accuracy of our predictors, we included additional contextual information, such as distance between places, GSM cell towers, WLANs or accelerometer data, but we observed no improvement. We also implemented various preprocessing techniques, such as clustering and feature embedding, with no improvement either.

4. BLENDING

As expected, the accuracies of the predictors presented in Section 3 are different, but more importantly, each might make different errors: samples for which a predictor fails may be those on which another excels. This idea is the foundation of blending, in which we combine several predictors, in order to take advantage of their diversity. We introduce several blending strategies in this section.

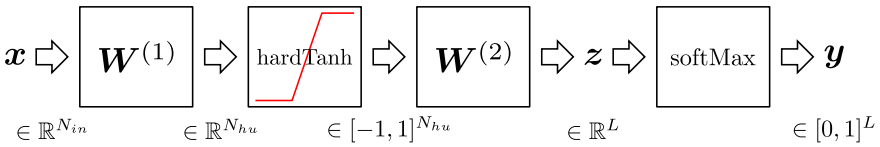

Figure 5: Architecture of our 2-layer ANN. A non-linear transfer function is applied between the first and second layer, and a softmax function is applied to the output, to obtain a probability distribution over places.

Note however that, because of data sparsity, we do not propose sample-based blending techniques, *i.e.*, techniques that adapt their choice of predictors according to the features of each sample. Indeed, learning a sample-based blending strategy would require a reasonable number of samples, in order to be accurate and not to overfit. Unfortunately, the small number of training visits available for each user is not sufficient for this task.

Before describing each strategy, we first define Φ, the set of all predictors trained on the data. This set can be split into three subsets of predictors $\Phi = \Phi_{\text{DBN}} \cup \Phi_{\text{ANN}} \cup \Phi_{\text{GBDT}}$, where each subset corresponds to all predictors of one same family. For instance, Φ_{GBDT} is the set of all predictors using the GBDT method. A predictor ϕ is defined by its family, some internal parameters, and the data it was trained with. Thus, we can refer to the predictor ϕ in general, or to a specific predictor $\phi(u)$, that was trained using the data of a user u. We do not mention the dependence on u when it is obvious from the context.

Below, we briefly explain the final five strategies we have implemented:

1. For each user, we choose the predictor that has the best accuracy over *set C*:
$$\phi_1 = \arg\max_{\phi \in \Phi} A_C(\phi).$$
We are aware that this choice could lead to poor generalization, as we may overfit to *set C*. However, it could also yield good prediction accuracy in the case where *set D* is similar to *set C*.

2. For each user, we choose the predictor that has the best accuracy, averaged over *set B* and *set C*:
$$\phi_2 = \arg\max_{\phi \in \Phi} \{A_B(\phi) + A_C(\phi)\}.$$
This strategy would reward the predictors with good generalization.

3. For each user, we first select the best predictors of each family, averaged over *set B* and *set C*:
$$\phi_{\text{DBN}} = \arg\max_{\phi \in \Phi_{\text{DBN}}} \{A_B(\phi) + A_C(\phi)\},$$
$$\phi_{\text{ANN}} = \arg\max_{\phi \in \Phi_{\text{ANN}}} \{A_B(\phi) + A_C(\phi)\},$$
$$\phi_{\text{GBDT}} = \arg\max_{\phi \in \Phi_{\text{GBDT}}} \{A_B(\phi) + A_C(\phi)\}.$$
Then, we simply combine these three predictors uniformly:
$$\phi_3 = \frac{1}{3}\phi_{\text{DBN}} + \frac{1}{3}\phi_{\text{ANN}} + \frac{1}{3}\phi_{\text{GBDT}}.$$

4. For each user, the predictor is a weighted mixture of all predictors, where the weight of each predictor is proportional to its average performance over *set B* and *set C*:
$$\phi_4 = \frac{1}{K} \sum_{\phi \in \Phi} (A_B(\phi) + A_C(\phi)) \cdot \phi,$$
where K is a normalizing factor to ensure the stochasticity of ϕ_4.

5. We choose the predictor that has the best average accuracy over all users for *set B* and *set C*:
$$\phi_5 = \arg\max_{\phi \in \Phi} \sum_{u \in \mathcal{U}} (A_B(\phi(u)) + A_C(\phi(u))),$$
where $\mathcal{U}$ is the set of all users, and $\phi(u)$ corresponds to the predictor ϕ trained using the data of user u, as explained above.
Note that, contrarily to the others, this strategy choose the same predictor for all users.

Results. We performed empirical tests with some of the blending strategies which showed improvements over individual predictors. However, because of our choice of data separation, we would need to have access to *set D* in order to assess the final performance of our blending strategies.

5. CONCLUSION

In this paper, we present various mobility predictors for the Nokia Mobility Data Challenge. We use a wide range of techniques, including Probabilistic Graphical Models and Artificial Neural Networks. Moreover, we adapt these techniques to the characteristics of the data, by implementing various mechanisms that ensure the adaptability of the predictors to the sudden changes in users' behavior and the sparsity of the data. In order to benefit from the diversity of these predictors, we introduce several blending strategies, that combine them into a global and more accurate predictor.

Our predictors reach an average prediction accuracy of more than 60%, yet we observe a high variance between users. Is this unpredictability mainly rooted in the users' personality, or is it is a consequence of data sparsity and noise?

Acknowledgements

We would like to thank Prof. Matthias Grossglauser and Prof. Patrick Thiran for the insightful discussions and their feedback about this paper.

6. REFERENCES

[1] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1st ed. 2006. corr. 2nd printing edition, Oct. 2007.

[2] R. Collobert. Torch. NIPS Workshop on Machine Learning Open Source Software, 2008.

[3] J. H. Friedman. Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 38:367–378, 1999.

[4] J. K. Laurila, D. Gatica-Perez, I. Aad, J. Blom, O. Bornet, T.-M.-T. Do, O. Dousse, J. Eberle, and M. 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.

[5] Y. Le Cun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural Networks, Tricks of the Trade*, Lecture Notes in Computer Science LNCS 1524. Springer Verlag, 1998.