

Delft University of Technology
Master's Thesis in Computer Engineering

Adaptive sampling with strict energy constraints

Pim Pjotter Nijdam



Adaptive sampling with strict energy constraints

Master's Thesis in Computer Engineering

Embedded Software Section
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands

Pim Pjotter Nijdam
pim.n@xs4all.nl

18th May 2017

Author

Pim Pjotter Nijdam (pim.n@xs4all.nl)

Title

Adaptive sampling with strict energy constraints

MSc presentation

30th May, 2017

Graduation Committee

prof. dr. K.G. Langendoen	Delft University of Technology
dr. R.R. Venkatesha Prasad	Delft University of Technology
dr. R.C. Hendriks	Delft University of Technology

Abstract

With more and more battery-operated devices equipped with sensors, it is becoming increasingly relevant to focus not only on energy efficiency, but also on ensuring that the available energy is not spent before the battery will be recharged. In this thesis a strategy for adaptive sampling is proposed that adapts the sample rate while respecting a predefined limit on the number of samples per discharge period. The proposed sample strategy aims to sample with uniform prediction errors, as opposed to uniform intervals. This signal-agnostic strategy is tested upon data from traffic flow, a wireless sensor node, and a position trace obtained with a mobile phone. A reduction in the root mean squared error ranging from 0 to 50 percent compared to uniform sampling was achieved.

Contents

1	Introduction	1
1.1	Problem statement	2
1.2	System overview	3
1.3	Methodology	4
1.4	Organization	4
2	Related work	5
2.1	Adaptive sampling	6
2.1.1	Offline methods	6
2.2	Compressed sensing	7
2.3	Predictors	7
2.4	Inspiration from information theory	8
2.5	Conclusion	8
3	Problem analysis	9
3.1	Energy budget	9
3.2	Predictability in a signal	10
3.3	Measure information	11
3.4	Predicting the variation in the prediction error	11
3.5	Sampling theorem	12
3.6	Putting it all together	12
4	Design	13
4.1	Algorithm	13
4.1.1	The setpoint	15
4.1.2	Update relative sample rate	16
4.1.3	Normalize relative sample rates	17
4.1.4	Calculating the sample rate	17
4.1.5	Condition transitions	17
4.1.6	Combining parameters γ and α	18
4.1.7	Initialization	18
4.1.8	Dealing with unknown relative frequencies	19

5	Simulation on Artificial Data	21
5.1	Simulation: Learn	21
5.1.1	Setup	21
5.1.2	Results	22
5.2	Simulation: Relearn	25
5.2.1	Setup	25
5.2.2	Results	25
6	Results With Real-World Data	27
6.1	Datasets	27
6.2	Condition: Hour of day	28
6.3	Condition: Connection type	34
6.4	Robustness	37
6.5	Optimality	38
7	Conclusions and Future Work	41
7.1	Conclusions	41
7.2	Future work	42
	Bibliography	44
A	RMSE prediction error	49

Chapter 1

Introduction

More and more devices measure signals about people or their environment. These devices (e.g. smart phones and wireless sensor nodes) are typically powered by (rechargeable) batteries [1] or perform some kind of energy harvesting [2, 3]. To obtain information, the battery simply should not be depleted before it will be recharged or the measurement period is over. The energy budget available for sampling poses an *energy constraint* on the sampling strategy.

For some applications, a certain quality of data is required. For others, the goal is to collect as much information as possible. *Uniform sampling*, i.e. sampling with a fixed time interval between samples, is probably the most widely used sampling strategy. Signals collected using uniform sampling are:

- easy to interpret; e.g. to calculate the probability a signal is above a threshold, the number of samples above that threshold can simply be counted and divided by the total number of samples.
- and easy to collect; only a periodic timer is needed to sample. To be able to accurately reconstruct the signal from uniform samples, the signal has to be sampled with at least the Nyquist-Shannon rate [4] (double the frequency of the bandwidth of the signal). Besides the bandwidth of the signal, no other information is required. Thus, uniform sampling is a *signal agnostic* sampling strategy.

However, uniform sampling is not necessarily an efficient sampling strategy in terms of energy usage. Sometimes the same amount of information can be gathered with fewer samples. For example, when tracking the position of an object, when the object does not move there is no need for measurements. Varying the sample rate over time is called *adaptive sampling*. Adaptive sampling strategies have been used to reduce power consumption [5, 6, 7, 8].

Often adaptive sampling is done by modelling the system and exploiting some knowledge about the system. For example, an accelerometer can be used to detect whether an object is motionless, and therefore does not change

in position. Tracking position is but an example, but it is a real-world problem that is easy to visualize and reason about. Therefore it is used as the primary example in this thesis.

Using adaptive sampling with a constraint on energy makes the problem more complicated. For the tracking example, the energy should be used during the time the object is moving. The power, and hence sampling rate¹, will be different depending on if the object moves a total of 10 minutes or 10 hours during the full period the object is tracked. Not knowing this in advance is what makes adaptive sampling with a constraint on energy more complex.

Continuing with the object tracking example, an accelerometer is used to determine when the object moves. But what if the relation between the accelerometer and movement is not so strict and the probability of movement is low, but not zero when the accelerometer measures no movement?. What should the sample rate then be? Also for other contextual information that might be used to adapt the sample rate (e.g. whether it is day or night), but for which the relation with the signal is not known a priori, can this contextual information be used to adapt the sampling rate?

In this thesis an adaptive sampling strategy is proposed that works with strict energy constraints, and learns to apply contextual information without having extensive knowledge of the signal a priori.

1.1 Problem statement

The problem addressed in this thesis is to develop a sample strategy to obtain as much information as possible from a signal, by sampling, while respecting a strict energy constraint.

The requirements for such a sample strategy are:

- adhere strictly to the energy constraint,
- be agnostic of the signal, and
- work on typical devices that perform sampling on signals related to people and their environment. That is, some reasonable limits should be posed upon the computational and memory requirements.

It is essential to adhere to the energy constraint. Without power no more samples can be taken.

For specific applications, adaptive sampling strategies are used that improve power efficiency. Can we design a sampling strategy that works without extensive modelling of the signal beforehand? Instead of modelling the signal and designing an adaptive sampling strategy a priori, such a sample

¹More samples usually correlates with higher power consumption. More information is provided in Section 3.1

strategy should learn to adapt to the signal. Besides being signal agnostic and thus ready to be deployed without extensive modelling of the signal, such a sampling strategy could exploit patterns in the signal that are very specific to the signal. For example, some objects might travel only during day-time and others only during night-time. An adaptive sampling strategy that learns this pattern could work well in both cases, without needing to know a priori which is the case. Of course, if this information is available, an application specific sampling strategy that doesn't need to learn this, but knows it a priori could work even better.

To work on typical devices, like wireless sensor nodes and smartphones, there are some limits on the computational and memory requirements. Even if the device is capable of performing heavy computations, they might use a lot of power.

1.2 System overview

Instead of sampling with a uniform time interval, the proposed system aims to sample with uniform prediction errors. To this end, the system incorporates a predictor that predicts future samples. This predictor can be as simple as predicting the signal to stay the same, or any arbitrarily advanced predictor. The system determines the prediction error from the predicted value and the true measurement. The prediction error during different conditions is tracked, e.g. during night and day, or when an accelerometer measures movement or not. The system can choose a different sample rate per condition. For example, sampling twice as often during the day than during the night. These conditions can be arbitrarily chosen. This allows usage of a wide variety of conditions, which in turn allows to exploit a wide range of patterns. For example, using the hour of the day a day-time pattern can be exploited, an immobility condition could be used to sample less during immobility, or any other discrete condition such as whether or not a system is connected to a wifi network.

The controller aims for a constant prediction error during all conditions. For example, with the conditions night and day, the prediction error should be the same during night and day. During conditions where the prediction error is above average, the controller decreases the sample rate. And during conditions where the prediction error is below average, the sample rate is increased. Thus the controller, which has its setpoint set to the average prediction error, reduces the variation in the prediction error.

The rate at which the controller adapts is controlled by a single parameter. Furthermore, a minimum and maximum sample rate can be set to keep the sample rate within specified bounds.

1.3 Methodology

The proposed sampling strategy was implemented using Python in a Jupyter-notebook [9] (formerly known as IPython-notebook), a notebook for interactive data science and scientific computing. Simulations with the sampling strategy were tested on artificially created datasets. This gave insight into the behaviour of the system and was used to check if the system works as intended. Finally, simulations were run on three real-world datasets: Indoor temperature readings of a wireless sensor node [10]; traffic flow measurements [11]; and a position trace, collected on a smartphone for this thesis.

1.4 Organization

The thesis is organised as follows. First an overview is given of related work and an analysis of the problem of sampling with energy constraints. (Chapters 2 and 3). Next, Chapter 4 follows with a mathematical description of the proposed system. To give insight on the behaviour of the controller, experiments on artificial data are documented in Chapter 5, followed by experiments on three data sets of real-world data in Chapter 6. Finally, conclusions on the proposed system are given in Chapter 7.

Chapter 2

Related work

It is quite common to have an energy constraint while collecting data with battery-operated devices or devices that harvest power. Some systems have a specific application, and hence are designed to strictly meet power and application demands. Other systems are designed to collect as much information as they can, which is typical when data is collected for analysis. For example, Eagle and Pentland collected data on student behaviour and interactions using mobile phones [1]. Using the collected data Eagle and Pentland used principal component analysis to discover structure in the data that they used to characterise and predict student behaviour [12]. To collect data for a study such as this one, the more data the better, and the energy constraint is provided by how often subjects are willing to recharge their phone.

Some strategies reduce energy consumption but do not adhere to a strict energy constraint. They aim at either one or both of the following:

- minimise energy consumption; done to extend battery lifetime with a sampling strategy that is considered to work well enough for the application
- keep the error within application-defined bounds; for example, tracking a device with a 1km precision, or for geofencing where only a high accuracy is needed close to a fence.

They can work well in reducing energy consumption, but given an energy constraint do not perform any better than uniform sampling. For example, not sampling when the accelerometer senses immobility is a useful optimization [13], but on its own can only save energy. To spend that saved energy during mobility, it would need to know how much energy is saved by immobility.

Keeping the error within application-defined bounds is a scenario for when an error bound can be determined a priori that would not result in violating the energy constraint. With strict error bounds the device could run out

of energy when more energy is needed to adhere to the error bound than is available. In that case the measurements stop, which could be fine for some applications as some good data is collected. We focus instead on applications where it would be better to measure with lesser accuracy than to stop sampling all together.

Minimising energy consumption and keeping the error within application defined bounds are not sufficient for our purposes: dealing with an energy constraint.

2.1 Adaptive sampling

Several adaptive sampling strategies can be found in literature in relation to the topics of collecting data with a smartphone or a WSN. For smartphones, in the context of collecting data, most power is typically consumed by gps [14, 5] and by transmitting over a mobile network [15]. Therefore many sampling strategies for smartphones are aimed at reducing energy spent to track the device's position. Examples of algorithms specifically designed for tracking position on a smartphone are Entracked [16] and EnLoc [6]. Entracked needs an application-defined error bound. Whenever there is no movement detected by the accelerometer no position update is measured, and if movement is detected the sample interval depends on the speed and the inaccuracy of the measured position. EnLoc predicts the user's location by predicting the route the user followed most often in the past, or the most followed route in the population and samples at positions where paths branch into multiple paths.

Law et al. [17] predict future samples for a wireless sensor network (WSN) node using an autoregressive integrated moving average (ARIMA) model and reduce energy consumption by skipping samples that are predicted to be within an application-defined error bound. The method relies on a pre-defined error bound to be able to adapt the sampling rate, whereas we aim to design a sampling strategy where no error bound needs to be defined a priori.

2.1.1 Offline methods

Some methods calculate a sampling strategy offline, using information from an earlier collected dataset.

OptiMos [18] looks at several methods to determine the best sampling distribution offline, in the context of placing mobile sensors on buses and determining the best times to sample, balancing information gathering and energy. Yan et al. collected data, determined a representative day of data and performed an offline search to determine a sampling distribution.

Jigsaw continuous sensing [7] includes an interesting technique to adaptively sample the gps. It uses the remaining battery budget, time and the

detected mobility (using an accelerometer) to look-up the sample rate, which has been calculated offline on a representative dataset using a markov decision process. The main disadvantage is that the algorithm needs to be trained and does not adapt to different mobility characteristics.

2.2 Compressed sensing

In 2006 Candès et al. and Donoho published two seminal papers [19, 20] that started the field of compressed sensing (CS). CS requires a base in which the signal can be sparsely represented (i.e. only few values that are not close to zero). For example, for images the base could be cosine functions, as many images can be sparsely encoded as a combination of cosines, something image compression algorithms heavily exploit. As long as the signal is indeed sparse in this base, CS can compress the signal.

CS has been applied in a number of cases to lossy compress measurements. For example to compress an ECG [21].

Besides compressing the sampled signal, with CS it is actually possible to take fewer samples, although this is non-trivial. To take fewer measurements it is not sufficient to just take a subset of the measurements in the original non-sparse base, but the measurements have to be performed differently. For example, Duarte et al. [22] designed specific hardware to apply compressed sensing directly while capturing a photo. The crux of their method is that, using a mirror, a single measurement (a sample of a photo diode) is the weighted sum of multiple pixels across an entire image.

Another example where CS can be used is to speed up magnetic resonance imaging (MRI) by taking fewer measurements [23]. Importantly, MRI measures frequencies, not pixels, and the resulting images can be sparsely represented in the frequency domain, and hence also sparsely measured in the frequency domain.

To conclude, CS is a very promising field, but not applicable for signals where a sparse base is unknown, and more importantly, for many applications it requires a new way of measuring.

2.3 Predictors

As explained in Section 1.2, the sample strategy presented in this thesis aims to sample with uniform prediction errors. To predict the signal, predictors ranging from simple to very complicated can be used. General predictors for time-series are exponential smoothing and Holt-Winter double exponential smoothing [24] for data that contains a trend. For stationary signals ARIMA methods have been used, for example Williams et al. predicted traffic flow using an ARIMA predictor [25]. For specific applications specific predictors

can be used, such as the predictor in EnLoc [6] that learns habitual routes to predict the position of people.

2.4 Inspiration from information theory

The problem statement is formulated in terms of gathering information. Hence, a measure for information is a very useful thing to have. In the field of information theory *entropy* is that measure. Mutual information is a measure that allows quantifying how much information one variable contains about another, it is more general than correlation and could for example be used to quantify how much information about the next sample is contained within the previous sample.

Despite the elegance of these measures, using entropy was found to be non-trivial and abandoned in favour of using a prediction error, which explicitly relies upon a predictor. The idea of sampling with a uniform prediction error was actually inspired from information theory, esp. the principle of maximum entropy [26]. In the case of sampling, one can apply the principle of maximum entropy to find the subset of samples that contains the most information.

2.5 Conclusion

To the best of our knowledge, no adaptive sampling strategies have been proposed that adhere to a strict energy constraint and are signal agnostic in the sense that they do not need extensive knowledge of the signal a priori.

Chapter 3

Problem analysis

The goal is to obtain as much information about the signal as possible using the available energy budget. Before addressing how to solve the problem a better understanding of 'as much information as possible' and 'available energy budget' is needed. In this chapter the problem will be formulated in such a way that we can give meaningful answers to those questions and gain some insight into the characteristics of the problem in a real-world setting.

3.1 Energy budget

Any active device without power is useless. Hence the energy budget, the amount of power available for measurements, is a hard requirement. Power consumption is a complex issue influenced by multiple factors. Yet, often power consumption can be expressed per sample, as a function of the power to perform the measurement and the power to wake-up the processor performing the measurement. Therefore the energy budget is modelled by the number of available samples.

3.2 Predictability in a signal

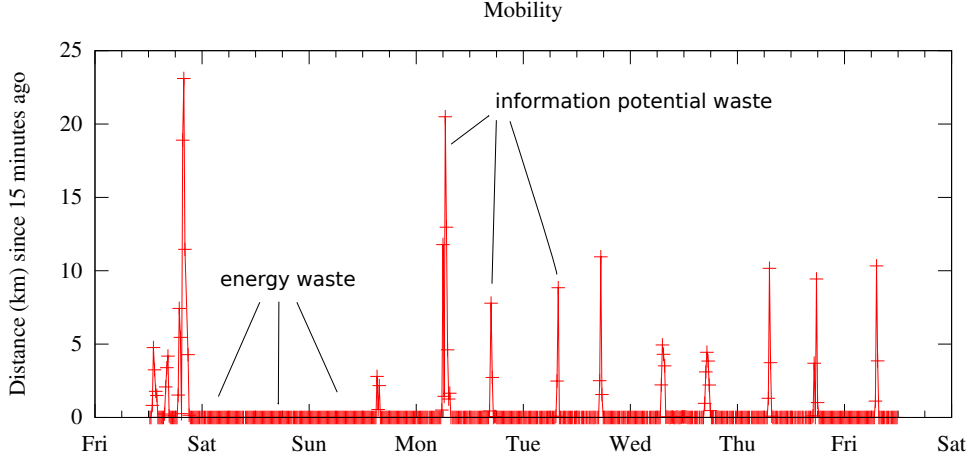


Figure 3.1: My mobility trace over a week.

Figure 3.1 shows travelled distance over a week, with the days of the week on the x-axis and the travelled distance on the y-axis. This trace is collected by fixed interval sampling and serves as an example. The trace starts with some travelling by bike, then Friday evening a long-distance travel, virtually no displacement during the weekend, a return trip Monday morning and then travelling to and from work the remaining days of the week. The two smaller peaks at Wednesday are due to commuting by cycle instead of public transport.

What is most noticeable, is that travelling occurs in bursts and is fairly regular. Of course this is not unique for this trace, but is apparent in most human location traces [27]. The samples obtained during these bursts contain a new position, whereas the samples obtained at other times merely contain the information that there was no displacement at all. Intuitively it is apparent that those samples obtained during bursts contain more information than others since it would be much harder to predict a missing sample during a burst than it would be to predict one of the other samples (e.g. just predicting a sample will be the same as the previous sample works well for most samples, but not during bursts).

In the trace of Figure 3.1, where a fixed sampling interval is used, most energy is spent (wasted) to obtain samples that contain little information (no displacement). We could improve the total amount of information contained in the trace by sampling more often during bursts and less when there is no displacement. In other words, we should exploit the variation in predictability of the signal.

3.3 Measure information

As in Figure 3.1 the displacement/change between two samples could be a measure for the amount of information obtained by the sample.

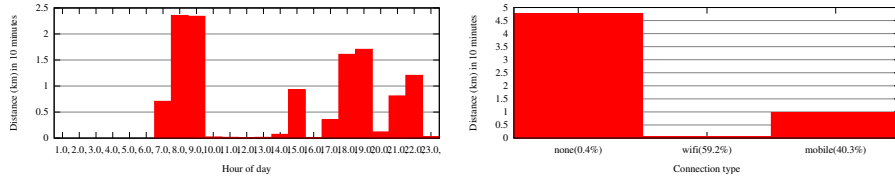
Considering the root mean square error (RMSE) . Defined as

$$RMSE(f(t)) = \sqrt{E \left[\left(\hat{f}(t) - f(t) \right)^2 \right]} \quad (3.1)$$

With $\hat{f}(t)$ the approximation or prediction of the signal. To capture the idea of displacement as a measure for the amount of information, the prediction can simply be the previous sample. In other words: a step function. In this case the *prediction error* and *displacement* are the same. But the term prediction error can be used with more advanced predictors. See Appendix A for an elaboration on this topic.

3.4 Predicting the variation in the prediction error

As seen in Figure 3.1 the prediction error varies a lot. Sometimes there is no displacement, while at few times there is a lot of displacement. To a certain extent this is predictable. See Figure 3.2b and Figure 3.2b where this variation is explained using the hour of day as conditional variable in the first figure, and the connection-type (phone connected to wifi, mobile or no connection) is the conditional variable in the second figure.



(a) The hour of the day and distance travelled show a strong dependency. (b) Connection type and distance travelled also show a strong dependency.

Figure 3.2: Mobility trace over a week.

Note that both figures show a pattern that is specific for this signal and will not generalize to the location trace of other people. What is more, it might not even generalize to other periods of time. For example, changing office hours will change the pattern observed for the hour-of-day condition and travelling on a train with a wireless connection affects the connection-type pattern. Even if not generalizable, some specific pattern is expected for each specific signal and can be exploited as long as the pattern is learned for each individual signal.

3.5 Sampling theorem

It is worth referring to the sampling theorem. A signal with bandwidth W can be specified by $2TW$ samples. With T the time. A relevant quote from Shannon [4]

The $2TW$ numbers used to specify the function need not be the equally spaced samples used above. For example, the samples can be unevenly spaced, although, if there is considerable bunching, the samples must be known very accurately to give a good reconstruction of the function. The reconstruction process is also more involved with unequal spacing.

Hence, with an adaptive sampling rate the signal can still be reconstructed for the same bandwidth. Although perhaps compromising somewhat on reconstruction accuracy. It is interesting to note that although bandwidth over the whole signal is not affected, bandwidth over part of the signal is. That is, during a burst of samples the burst can be reconstructed with a higher bandwidth.

3.6 Putting it all together

To adapt the sample rate, intuitively it seems that we need no (or very few) samples when the signal does not change, and as many as possible when the signal does change. In case of a little change versus a lot of change it is less obvious what the exact sample rates should be. We can try to reduce the variation in displacement such that each sample incurs the same displacement. That way a position would be sampled every x meters instead of every t seconds. The distance between each sample would be the same. Or more generally, for any predictor (not just using the last sample), the magnitude of the prediction error for all samples would be the same. This can reduce the RMSE of the signal, however it does not minimise the RMSE of the signal, at least not optimally. However, it does seem a promising heuristic. Thus reducing the variation in the prediction error is used as the main idea to implement adaptive sampling.

Chapter 4

Design

This chapter describes the algorithm that adapts the sample rate with the goal to reduce the variation in the prediction error.

A discrete variable is chosen that is expected to be a statistically dependent variable for the prediction error. This variable is called the *condition variable*. For example, each hour of the day can be a condition. For each condition there is a sample rate. The controller will try to reduce the variation in the prediction error by adjusting the sample rate during each condition such that the prediction error during each condition moves towards the mean prediction error. The mean prediction error is set as the setpoint of the controller. The prediction error is chosen to be the difference between a sample and previous sample, however the algorithm will work with any other predictor.

4.1 Algorithm

Here follows a description of how the system works. Table 4.1 contains a list of used symbols and is mainly meant for reference.

To allow the system to vary the sample rate, it adjusts the relative sample rates. The relative sample rate $r_{c,i}$ is a vector with multipliers for the sample rate. For condition c , the sample rate of the system is the product of the corresponding relative sample rate and the reference sample rate. At the beginning all multipliers are set to 1, resulting in a fixed sample rate. As these multipliers are adjusted the system will use different sample rates during different conditions. These multipliers are normalized such that weighing the multipliers by the corresponding probabilities for each condition gives an average relative sample rate of 1 Hz.

$r_{k,i}$	Relative sample rate for condition k , at iteration i
C	set of conditions. $ C $ denotes the number of conditions
S_i	Sample at iteration i
$\hat{S}_i$	Sample prediction for iteration i
l_i	Prediction error (l for loss) at iteration i
L_i	Normalized and clipped prediction error at iteration i
c_i	Condition at iteration i
p_c	Probability of condition c
μ_i	Setpoint of the system at iteration i
t_{half}	Parameter to tune how fast the system adapts.

Table 4.1: Glossary of used symbols.

The goal of the algorithm is to reduce the variation of the prediction error. To do this during each condition the prediction error is measured and the relative sample rates are adjusted. The algorithm is a controller that adjusts the sample rate based upon the observed prediction error. To reduce the variation, the setpoint of the controller is set to be an average of the observed prediction errors for all conditions. To enable the controller to adjust to patterns in the prediction error that appear and disappear over time, the controller is biased toward recent prediction errors by using an exponential weighted average to calculate the setpoint.

This results in a controller looping over the following steps:

1. Observe the prediction error for a sample
2. Update the setpoint (low pass filter)
3. Adjust the relative sample rate for the current condition based upon the distance to the setpoint
4. Normalize the relative sample rates so they average to 1
5. Calculate the new sample rate (i.e. the time to the next sample)

As in most controllers, the rate of adaptation is controlled by a parameter, in this case t_{half} . This parameter is carefully crafted to ease tuning robustness. For example, the parameter is invariant to scale and bias of the signal.

The above steps are explained in more detail in the following sections.
prediction error

The overall goal of sampling is to obtain as much information from a signal as possible. As explained in Section 3.2, the amount of information contained in a sample is related to the predictability of the sample. The predictability of a sample can be quantified by the prediction error. A mathematical definition is given in Equation 4.1. Note that for this definition the prediction error can be positive and negative.

$$l_i = S_i - \hat{S}_i \quad (4.1)$$

With $\hat{S}_i$ the prediction of the sample value S_i . The last sample is usually a good prediction for the next sample. Therefore Equation 4.2 is used for the prediction error throughout the thesis, unless stated otherwise.

$$l_i = S_i - S_{i-1} \quad (4.2)$$

4.1.1 The setpoint

The prediction error during each condition is steered toward the setpoint, by adjusting the sampling rates for the conditions. As explained in Section 3.6, the goal of the algorithm is to reduce the variation in the prediction error. This is done by setting the setpoint to be the average prediction error.

Since the prediction error changes over time (it is non-stationary) an exponential moving average of the prediction error could be taken.

$$\mu_i = \alpha_c \cdot \mu_{i-1} + (1 - \alpha_c) \cdot l_i \quad (4.3)$$

This has a slight drawback, the average is biased towards the most recent conditions. Therefore, a small intermediate step is taken. An average of the prediction error is calculated *per condition* and then averaged over all conditions. This slight alteration introduces a little bit more overhead as for each condition an exponential moving average ($X_{c,i}$) is stored.

$$X_{c,i} = \begin{cases} \alpha_c \cdot X_{c,i-1} + (1 - \alpha_c) \cdot l_i & \text{if } c = c_i \\ X_{c,i-1} & \text{else} \end{cases} \quad (4.4)$$

Thus the setpoint is defined as

$$\mu_i = \sum_{c \in C} X_{c,i} \cdot p_c \quad (4.5)$$

4.1.2 Update relative sample rate

The observed prediction error is used to update the sample rate of the condition for which the observation was done. A multiplicative update rule is used, of the form:

$$r_{c,i+1} = r_{c,i} \cdot e^{\gamma \cdot (l_i - \mu_l)} \quad (4.6)$$

With γ a parameter that determines how fast the system adapts. The actual formula used is a variation that address three issues with the formula is this form

- the sample interval varies, but the formula should be invariant to the sample interval so that γ does not need changing
- the magnitude of the prediction error might be different for different signals and would not require the same sensitivity for updating the relative sample rate
- the function is extremely sensitive to outliers

These shortcomings are addressed with the actual formula

$$r_{c,i+1} = r_{c,i} \cdot e^{\gamma_c \cdot L_i \cdot \Delta t_i} \quad (4.7)$$

where the multiplication with Δt_i , the interval since the last sample, makes the rate of adaptation invariant with respect to the sample interval. To address the problem of different magnitudes for the prediction error, the prediction error is normalized. To address the sensitivity to outliers, the prediction error is clipped. Thus L_i , the normalized and clipped prediction error, expresses the number of standard deviations the prediction error deviates from the setpoint. Since the setpoint is set to be an average over the prediction error itself, one could think of L_i as a z-score or t-statistic. However, there is one significant difference: the prediction error goes through a low-pass filter before being calculated. (As mentioned before, the low-pass filter allows the algorithm to adapt to long-term temporal changes in the predictors performance). To calculate the standard deviation, the standard deviation of the (low-pass) mean prediction error over all conditions is used. L_i is defined as

$$L_i = \min \left(\max \left(\frac{l_i - \mu_i}{\sqrt{\sum_{c \in C} (X_{c,i} - \mu_i)^2 \cdot p_{c,i}}}, -k \right), k \right) \quad (4.8)$$

Finally, the normalized prediction error is clipped between k and $-k$. This is to inhibit extreme sensitivity to outliers. During experiments a value of 3 for k was used. 3 standard deviations is conservative.

Note that there is a parameter γ_c for each condition. This is just for ease of notation as the parameter depends on the probability of the condition. Conditions with the same probability have exactly the same parameter.

The resulting formula has an easy to understand parameter γ_c that determines how fast (in time) the algorithm tries to reduce the variation in the prediction error. An intuitive way of understanding this is to think of it as a half time, similar to the half time of radioactive material. γ_c determines how fast the system tries to half the variation in the prediction error.

4.1.3 Normalize relative sample rates

Constraints can be set on the individual relative sample rates. A minimum and maximum rate. A minimum sample rate can be required for a specific application. But also it makes sense as a safety-precaution to ensure the system cannot tweak the sample rates too extreme. The maximum rate is provided for completeness, but was not used during experiments. Note, that when only one constraint is given, automatically the other is implicitly determined. For example, with two conditions with 50% probability and a minimum relative sample rate of .5 for one condition, the limit for the relative sample rate of the other condition would be 2.

The relative sample rates should average to 1 so that overall the system takes the expected number of samples. To achieve this, after a sample rate is updated and the constraints satisfied, all relative sample rates are divided by the mean sample rate over all conditions ($\sum_{c \in C} r_{c,i} \cdot p_c / |C|$). One caveat: the normalization step can alter the constraint, but in practise, only very, very little. This has not been a problem.

4.1.4 Calculating the sample rate

The system calculates the actual sample rate as a product of the reference sample rate set for the system and the relative sample rate for the current condition.

$$R_i = R_{ref} \cdot r_{c_i,i} \quad (4.9)$$

4.1.5 Condition transitions

After a condition transition it makes sense to use the sample rate for the new condition. The sample rate is chosen to be a weighted average of the sample rate for the old condition and the new condition. In this case it is easier to think of it in terms of an interval. For example, the old interval is 3 minutes and the new interval is 12 minutes. If the condition transition occurred after 1 minute, one third of the interval has passed. For the remaining two thirds the new interval will be used, resulting in another 8 minutes.

Now what to do with the prediction error incurred for the sample? Due to the condition transition it is not solely contributed to either condition. But it should not be ignored. It could be attributed over the conditions by storing condition transitions that happened during the sample interval, although theoretically this could be many. This case is handled like all samples, the prediction error is simply attributed to the last condition.

4.1.6 Combining parameters γ and α

With a two dimensional search space finding good values for the parameters can be tricky. Both parameters roughly determine how quickly the system adapts. Parameter γ affects how quickly the system changes the sampling rates to converge the prediction error to the setpoint and parameter α affects how quickly the setpoint is adapted. Intuitively it makes sense that these two time related parameters (adapting the sampling rate and adapting the setpoint) are related. If the setpoint changes faster than the system adapts, this makes for an unstable system. On the other hand, if the systems adapts much faster than the setpoint is learnt, the system is adapting to an *old* setpoint. Hence, to simplify the system both parameters are combined into a single intuitive parameter t_{half} . From t_{half} γ_c and α_c are calculated as follows

$$\begin{aligned}\gamma_c &= \frac{\ln(2)}{t_{half} \cdot p_c} \\ \alpha_c &= \left(\frac{1}{2}\right)^{1/(t_{half} \cdot p_c)}\end{aligned}\tag{4.10}$$

The formulas are constructed so that for $t_{half} = 1$, $e^{\gamma_c \cdot L \cdot \Delta t} = 2$, the relative sample rate will be multiplied by 2, and $\alpha_c = 0.5$, the new sample is given half the weight for calculating the low-pass mean prediction error. Thus the relative sample rate and setpoint adapt at approximately the same rate. The factor p_c ensures the mean prediction error for each condition is updated equally fast, irrespective of the probability of the condition (without accounting for p_c , conditions with a higher probability would adapt faster than conditions that occur less frequent).

4.1.7 Initialization

To make sure the algorithm has enough representative data, an initialization phase is used. The initialization phase is a phase of configurable duration during which the algorithm only observes, but does not alter the relative sample rates (which start out as 1, a fixed sample rate). The prediction error is observed and the low-pass filters are fed data. After the initialization phase the update rule starts updating the relative sample rates.

4.1.8 Dealing with unknown relative frequencies

For the condition *hour of day*, the relative frequency is known a priori. But for some conditions this is not the case. Those conditions are much harder to use, since we do not know for how long the sample rate should be increased, and hence by how much it can be improved. An attempt has been made to deal with such conditions using the following techniques.

The relative frequency is estimated with the relative frequency of occurrence of a condition so far. Instead of multiplying the relative sample rate with the reference sample rate to determine the actual sample rate, the relative sample rate is multiplied by the remaining number of samples divided by the remaining time. If there is only one sample remaining, the remaining time is used as the sample interval. Furthermore, a quota is introduced for each condition. When more time is spend in a condition than predicted, instead of using the relative sample rate multiplier, the sample rate is simply set to be the remaining number of samples divided by the remaining time.

Chapter 5

Simulation on Artificial Data

Before studying the effectiveness of the algorithm on a realistic dataset, a few simulations on an artificially-constructed dataset were performed. The goals of these simulations is to test the algorithm and get insight into the behaviour of the algorithm for specific cases.

5.1 Simulation: Learn

The goal of this simulation is to study the behaviour of the system for one of the simplest cases. This case has the following two assumptions.

The first assumption: fixed known probabilities for the conditions. For this the time of day is used as conditional variable. To simplify analysis of the simulation the time of day is split into two conditions: night (am) and day (pm). Since each day consists of 12 hours of day time and 12 hours of night time both conditions have an equal a priori probability of 50%.

The second assumption: a single solution to the sampling rates that does not change over time. The pattern in the data does not change; the optimal sample rates for the two conditions remain the same over time. Hence the system needs to find the solution only once. This is implemented with a counter that increases at different rates during night time and day time. For example during the night the counter might increment by 1 each second, while during the day the counter might increment by 5 each second.

5.1.1 Setup

The signal is piecewise linear, with a slope of 0.005 during night time and a slope of 0.001 during day time. The overall sample rate is 1 sample every 4,000 seconds (01:06:45 h).

For this simulation only parameter γ , the learning parameter of the algorithm, is varied. γ is set to 0, 2, 5 and 40 to produce the graphs. As the signal characteristics do not change over time α is set to 1.

A technical note: the simulator calculates the error per 20 seconds. The simulator does not represent the artificial signal as a continuous signal, but rather as a discrete one. An interval of 20 seconds was used for performance.

5.1.2 Results

Each figure consists of three parts that share the x-axis representing time. The first plot is of the sampled signal. The space between samples reflects the sample interval.

The second plot is of the prediction error, in this case the difference between the taken sample and the previous sample. This plot also contains the setpoint for the controller.

The third plot is of the relative sample rates during the day time and night time. The blue line corresponds with the current condition (am or pm, depending on time of day) and the green line with the other condition. Therefore the crossings of the lines represent the change of current condition at noon and midnight.

Figure 5.1 shows the results for $\gamma = 0$, which means the system does not learn and uses a constant sample rate. This is the baseline experiment. Figures 5.2 and 5.3 show the same simulation but with different values for γ . As can be seen in both cases the sample rate converges towards near-optimum. The optimum being a prediction error of 0.6, realised with sample rates with a ratio of 1:5. Thus a sample rate of 1.67 during night time and 0.33 during day time. Setting γ too high however results in an unstable system, as illustrated in Figure 5.4 where the sample rates of the two conditions do not converge but are rather erratic.

This simulation shows that given a clean data set, for reasonable values for γ the system behaves as desired.

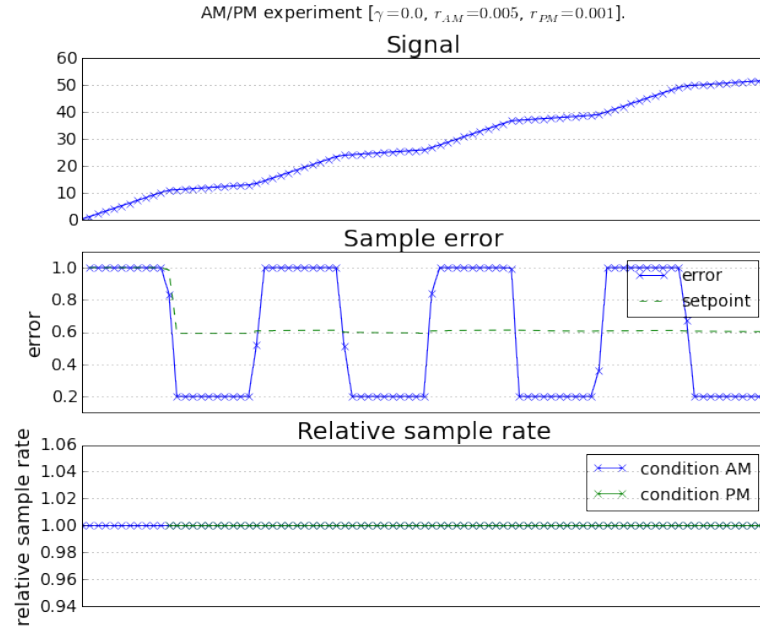


Figure 5.1: Baseline: With $\gamma = 0$ the algorithm produces a fixed sample rate.

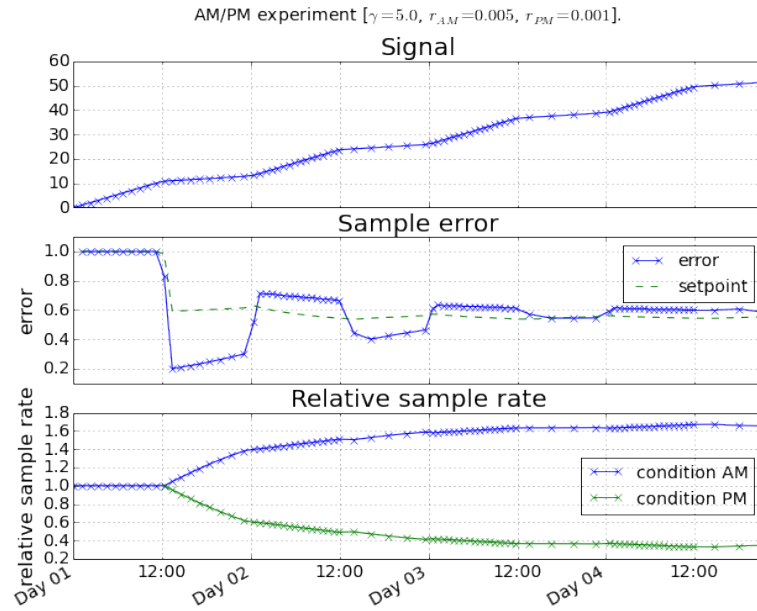


Figure 5.2: Learn: With γ hand tuned for illustration, the error converges towards the setpoint.

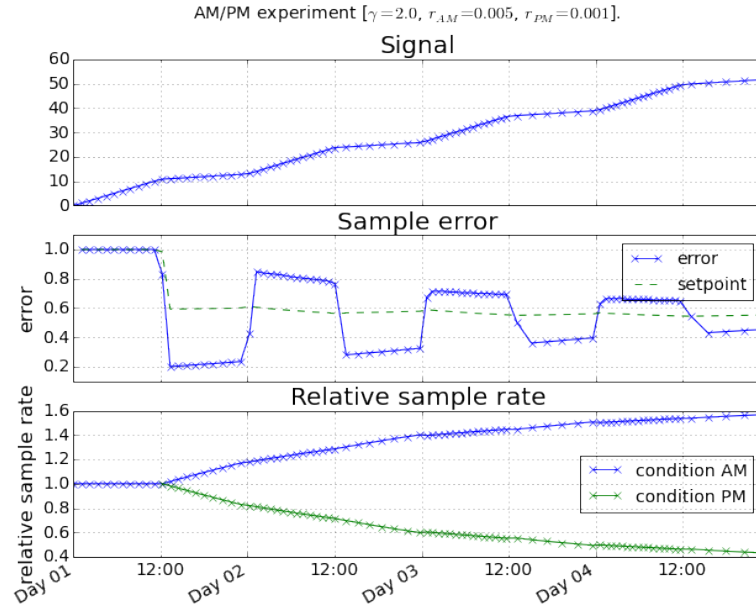


Figure 5.3: Learn: Decreasing γ yields slower convergence.

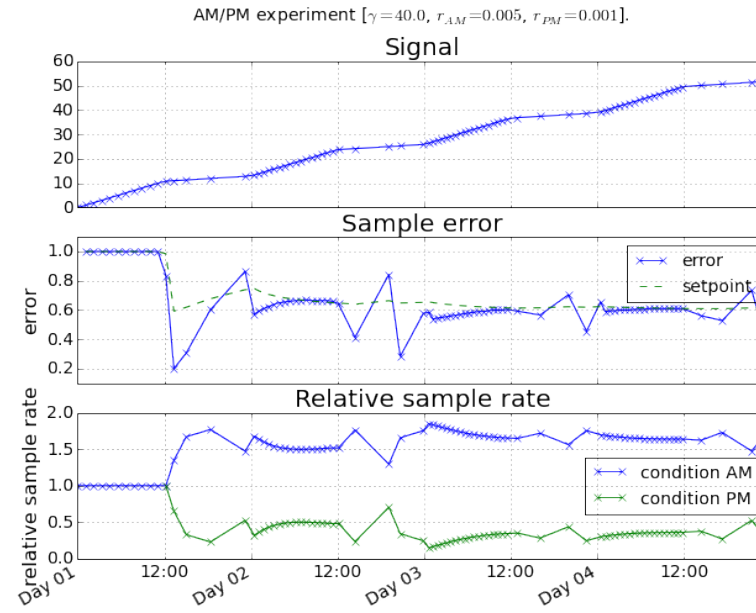


Figure 5.4: Learn: Setting γ too high makes the system unstable.

5.2 Simulation: Relearn

The goal of this simulation is to test the behaviour for a slightly more complex case than simulation 1, viz. the signal characteristics change and have to be relearned. In the simulation the system learns the signal characteristics during two periods, day and night. In this simulation the day and night characteristics of the signal will change halfway during the simulation.

5.2.1 Setup

The setup of this simulation is similar to simulation 1. But this time the signal will be 8 days instead of 4, for the four new days the slope of the signal during am and pm will be reversed. Thus the signal will have a slope of 0.001 during night time and a slope of 0.005 during day time.

Only parameter t_{half} will be varied during the simulation. t_{half} will be set to 12, 24 and 72 hours.

5.2.2 Results

Figures 5.5 - 5.7 show that the system relearns the new characteristics over time. By inspecting the graphs closely it can be seen that t_{half} affects the learning time roughly as expected.

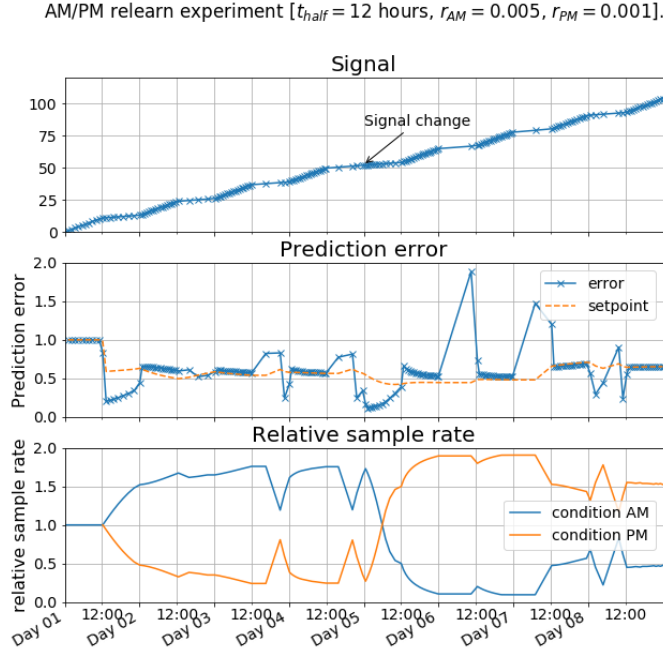


Figure 5.5: Relearn: A low t_{half} results in quick adaptation.

AM/PM relearn experiment [$t_{half} = 24$ hours, $r_{AM} = 0.005$, $r_{PM} = 0.001$].

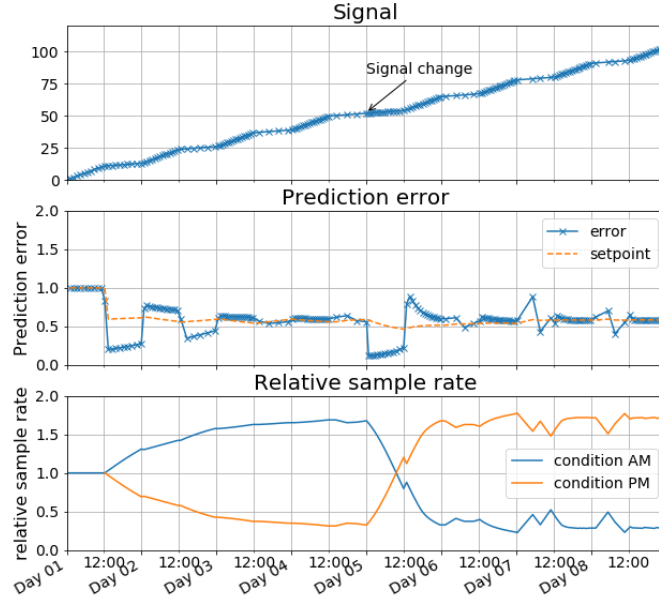


Figure 5.6: Relearn: $t_{half} = 24h$.

AM/PM relearn experiment [$t_{half} = 72$ hours, $r_{AM} = 0.005$, $r_{PM} = 0.001$].

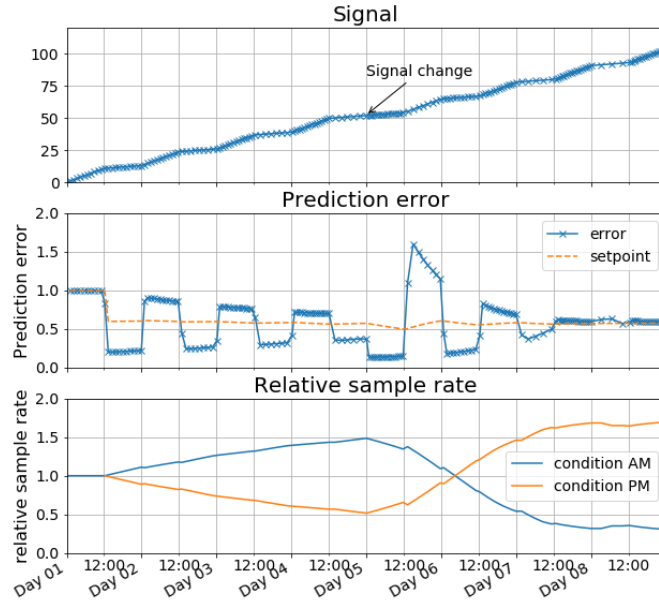


Figure 5.7: Relearn: Slowly adapting with $t_{half} = 72h$.

Chapter 6

Results With Real-World Data

The algorithm has been tested on several datasets. Section 6.2 contains the results on several data sets using the hour of day as conditional parameter. These results show some significant improvements the algorithm can yield using even such a simple conditional as this.

To measure the quality of the sampled signal, the RMSE is calculated as follows. The sampled signal is matched with the original signal so that for every sample in the original signal, there is a sample from the sampled signal. The sample at the same time is taken if available, otherwise the latest sample before that timestamp. The RMSE is calculated on the distance between the original samples and the obtained samples. For single values (temperature, etc.) arithmetic subtraction is used. For two position points, the great-circle distance between the points is calculated using the haversine formula[28].

6.1 Datasets

Datasets have missing values and irregular sample frequencies. During short gaps the sample rate cannot be increased, because there is no data available. Therefore some datasets were resampled. For each this is specified. Still, there are some gaps in the datasets. To handle these remaining gaps, the algorithm ignores the prediction error from samples where the gap is more than twice the intended sample interval.

RTA

The RTA dataset from the Kaggle competition RTA Freeway Travel Time prediction[11] contains approximately a year of measurements on Sydney's M4 freeway. Loops measure the number of cars crossing over every 3 minutes. Samples from loop 40010 are used. This dataset was chosen because of

the long duration over which data is available. Consistent data is available every 3 minutes on all weekdays, but no data during weekends.

Position trace

Since it is hard to find datasets over a long time (several weeks) with personal data, I have collected my own data. Over a duration of 24 days I've measured my position using an iPhone. Excluding gaps of more than 1 hour, the mean interval is 87 seconds with a standard deviation of 142 and a reported mean accuracy of 56 meters with a standard deviation of 22 meters. The data is resampled to a 1 minute interval. If there are multiple measurements available the mean is used, if there are no measurements available gaps up-to 1 hour are interpolated using linear interpolation.

Temperature

Intel Lab Data[10] contains measurements taken from a wireless sensor network of mica2 nodes installed at the Intel Berkeley Research lab. It is about three weeks of data on temperature and other sensors sampled every 31 seconds. There is a lot of missing data. Therefore the node with the most temperature readings (mote id 31) was selected. Data after 2004-03-22 is discarded, as the temperature readings are clearly incorrect. (consistent temperatures of over a 100 degrees Celsius). The data is resampled to a 1 minute interval. If there are multiple measurements available the mean is used, if there are no measurements available gaps up-to 1 hour are interpolated using linear interpolation.

6.2 Condition: Hour of day

Experiments were performed using the hour of day as condition variable. Hours are rounded down, so 16:50 becomes 16. Hour of day is one of the easiest conditions to use, as it is known for all signals with a timestamp. Furthermore, hours occur predictably and hence no need to guess the relative frequency of the condition. Last but not least, a lot of signals correlate strongly with hour of day. Physical measurements like weather, behaviour of animals and people, actually almost everything is influenced by our earth's rotation.

Parameters used by the algorithm are shown in Table 6.1. Since sampling with a higher sampling rate than the original signal simply cannot be done. $r_{\max}$ is set so that the algorithm never tries to samples with a higher sample rate than the original signal. The experiment is performed with a sample rate ranging from 5% to 25% of the original sample rate. Low values are chosen because there must be room to vary the sample rate, with a sample rate close to the original it cannot be increased much.

Parameter	value
t_{half}	1 week
t_{init}	2 days
r_{min}	$\frac{1}{4}$
r_{max}	original sample rate
clipping	5 (standard deviations)

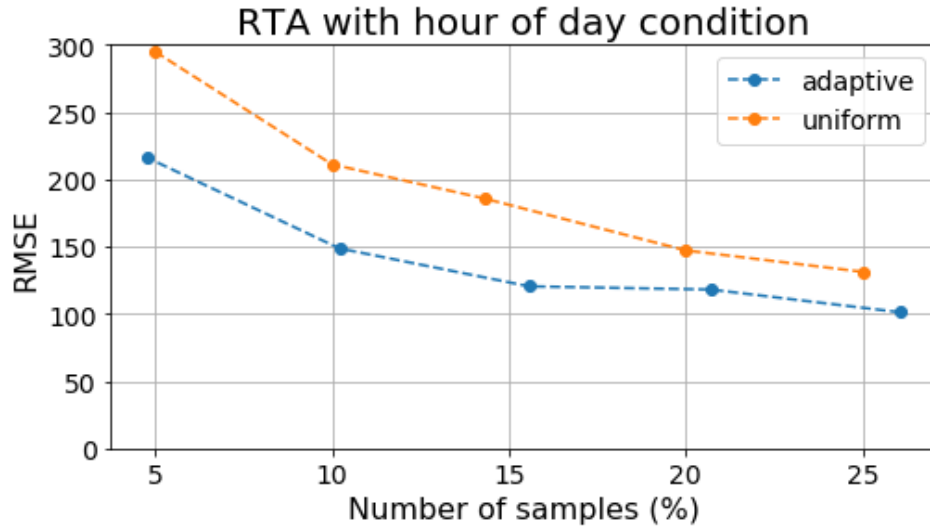
Table 6.1: Parameters used for the sampling strategy with hour of day as condition variable.

RTA

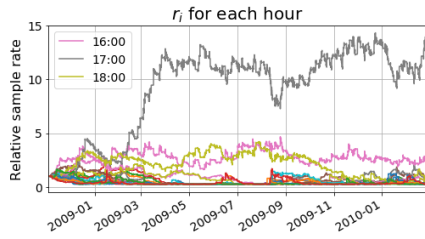
Figure 6.1 shows the results on the RTA dataset. As can be seen the RMSE with adaptive sampling is consistently much lower than the RMSE for uniform sampling, up to approximately 66% at 5% sampling rate. What is more, for 5% and 10% sampling rate the RMSE is approximately the same as uniform sampling with twice the number of samples.

In the data a trend can be observed, with fewer samples adaptive sampling performs relatively better than uniform sampling. One explanation might be that with fewer samples the sample rate can be varied more, with 25% the relative sample rate cannot be more than 4 due to a lack of high frequency data. As can be seen in Figure 6.1b higher relative sample rates are exploited by the algorithm. To investigate the extent to which this might explain the trend, an experiment was performed with clipping the maximum relative sample rate at a factor 4. The results are shown in Figure 6.2. As can be seen, clipping does decrease performance, but surprisingly not for the lowest sample rate (5%). This makes it likely there is another explanation for this trend. In fact, this trend also shows up during the other experiments. Reducing the number of samples improves the relative performance of adaptive sampling.

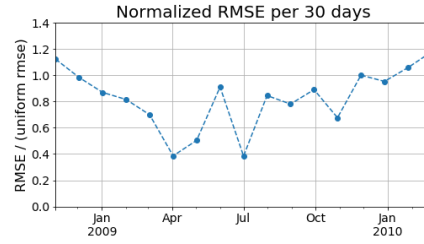
Figure 6.1c shows that the relative performance of adaptive sampling varies. Most months it performs better than uniform sampling, but not always. During the first month it clearly under performs.



(a) Adaptive sampling shows a clear improvement over uniform sampling.



(b) The variation of sample rates learned over time. This is with a 5% sample rate. As can be seen, the majority of samples are taken during a few hours.



(c) Normalized RMSE over time. Most of the months adaptive sampling improves the signal quality significantly.

Figure 6.1: Adaptive sampling on RTA data set using hour of day as conditional variable.

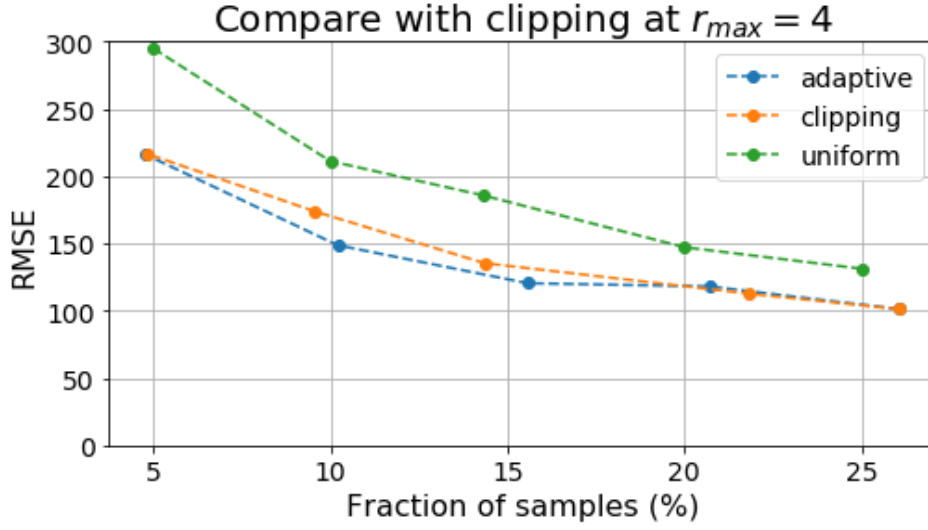
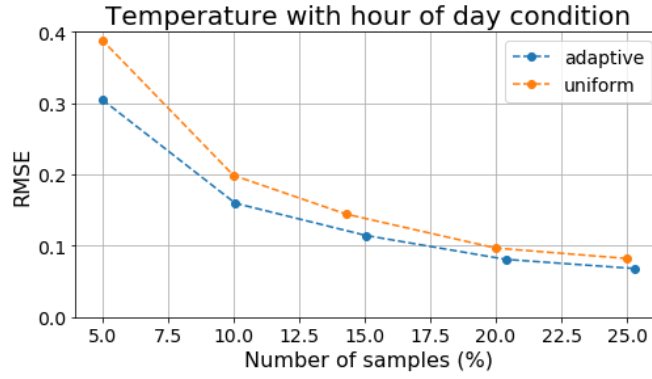


Figure 6.2: Clipping the maximum relative sample rate at a factor 4 does.

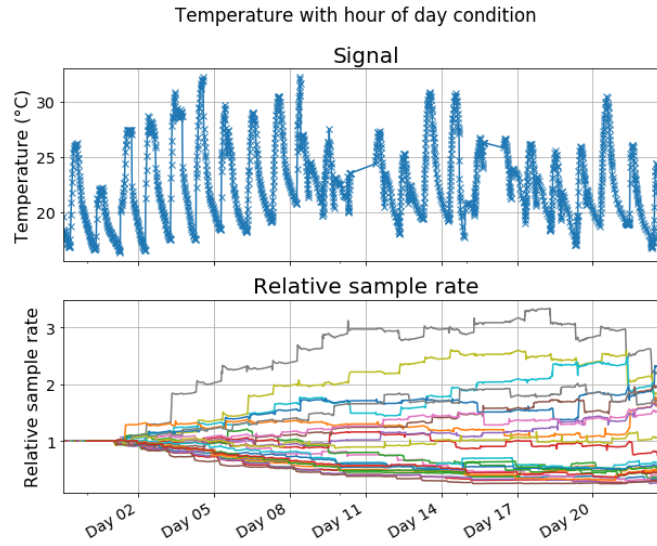
As can be seen, for 15% the uniform sample strategy has taken slightly fewer samples than the adaptive strategy. This is due to rounding. There is no original sample available at a rate of 15%, but the closest sample available is at a sample rate of 14.3%. Now why does the adaptive strategy take more samples? It varies the sample rate and it turns out that even when the same number of sample rates are rounded down as up, the rates that where rounded up add more samples than the rates that where rounded down. e.g. we try to resample a signal sampled at 1 minute to 1.5 and 1.499 minutes. Taking 30 minutes of data and rounding 1.5 to 2 and 1.499 to 1, we end up with 15 and 30 samples respectively, which is respectively 5 samples fewer, and 10 samples more than the expected $\frac{30}{1.5} = 20$ samples, in total adding 5 samples. This effect becomes less pronounced with higher sample rates, as the closest available sample rate is closer to the desired sample rate.

Temperature

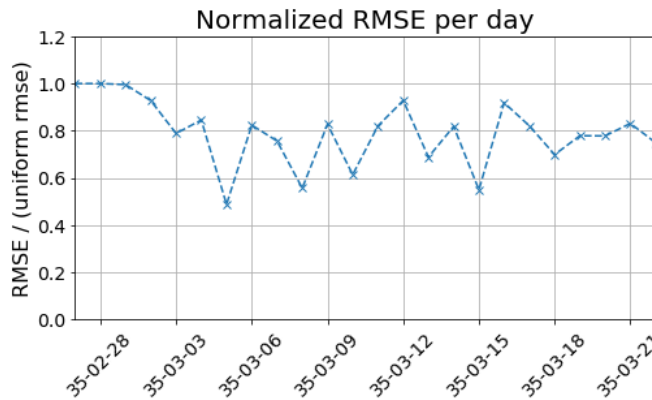
Figure 6.3 shows the results on the temperature dataset. As can be seen, the gains for adaptive sampling are less, though there is still a significant improvement over uniform sampling. There are two reasons. First, because the data set is relatively short the initialization phase has a bigger impact. Second, the signal changes are more evenly distributed over all hours during a day.



(a) Adaptive sampling shows a slight improvement over uniform sampling.



(b) The variation of sample rates learned over time, with a 5% sample rate, for the temperature dataset. The variation in sample rates is spread over many hours, as opposed to the RTA experiment (see Figure 6.1b) the variation in sample rate is mainly due to a few hours.

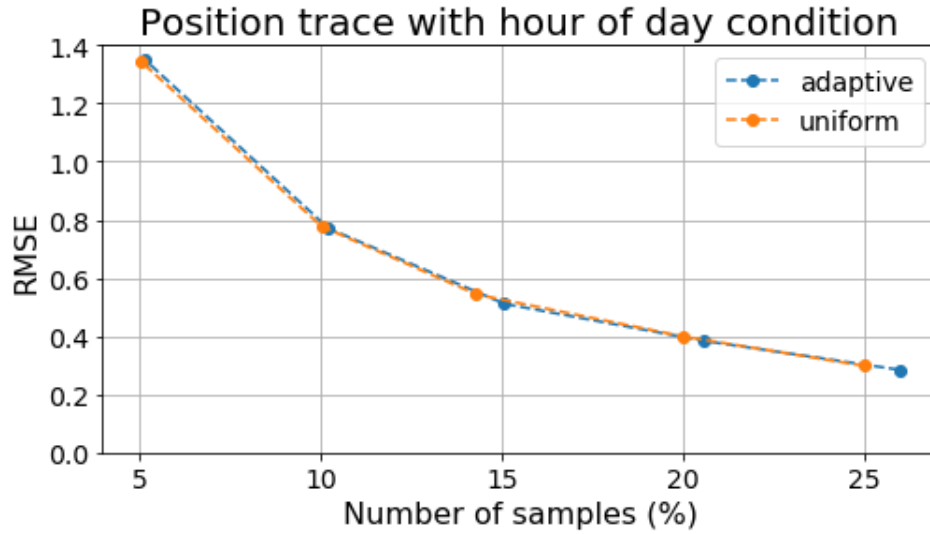


(c) The normed RMSE varies quite a bit. Showing that the relative gains from adaptive sampling vary a lot over time.

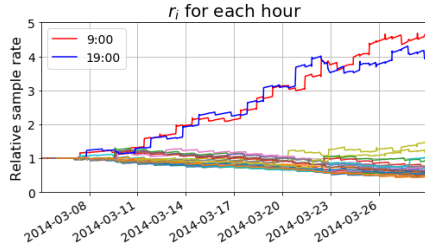
Figure 6.3: Adaptive sampling on the temperature data set using hour of day as conditional variable.

Position

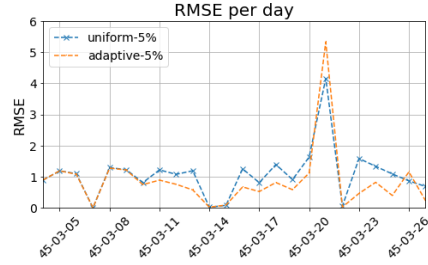
Figure 6.4 shows the result on the position data. As can be seen adaptive sampling perform very similar to uniform sampling. Upon closer inspecting of the RMSE per day in Figure 6.4c, it can be seen that adaptive sampling is actually consistently improving over time, but one day (the peak in the graph) it performs much worse than uniform sampling.



(a) Adaptive sampling performs similar to uniform sampling.



(b) The variation of sample rates learned over time. This is with a 5% sample rate. The two hours for which the relative sample rates increase steeply are the hours of commute to and from office.



(c) RMSE over time for adaptive and uniform sampling at 5% sample rate. There does seem to be a learning trend, as adaptive sampling consistently outperforms uniform sampling in the second half of the graph, with the notable exception of one huge peak that undoes the advantage.

Figure 6.4: Adaptive sampling on position data set using hour of day as conditional variable.

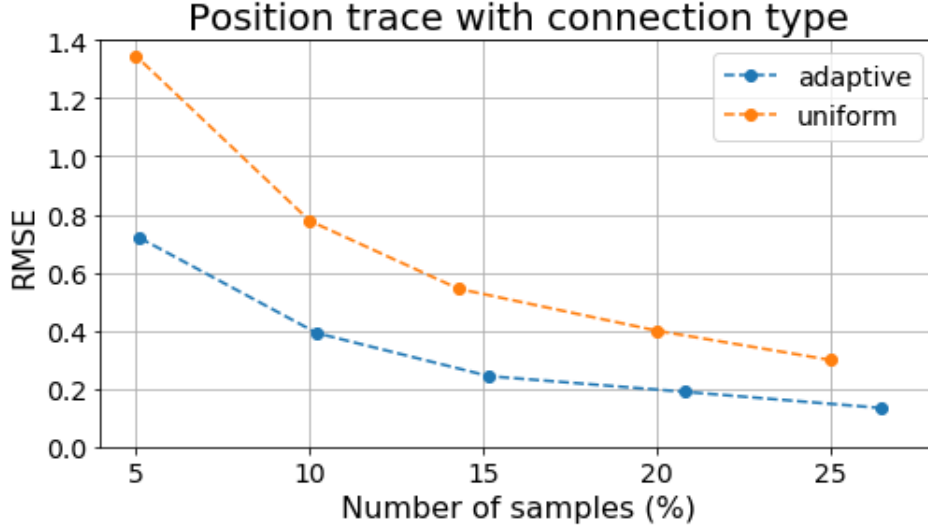
This is actually a very revealing experiment. Analysing the data more carefully we see that before the peak a pattern is apparent, a commute to

office and back during weekdays, and little travelling during the weekend. Then the pattern is broken by instead of travelling home after work, there is some travelling in the evening through the city, before eventually returning home. Exploiting the existing pattern actually performs worse than uniform sampling, during that day. The algorithm is learning the sample rate for each hour of the day independently, 24 different values, that is quite a bit to learn and might need a longer time to adapt well to the travelling pattern. In contrast, the RTA data set contains a year of data.

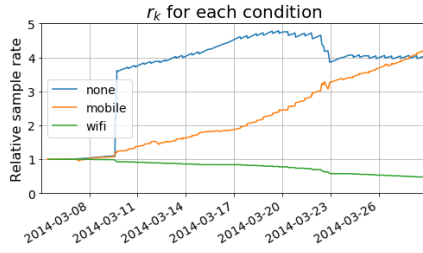
6.3 Condition: Connection type

Although hour of day can be an effective condition, other conditions might be exploited. When recording the position data with an iphone, a lot of other data was recorded as well. Among them, the phones connectivity. i.e. the phone either has no connection, a mobile data connection or a wifi connection. Since a connection to wifi usually means no travelling, this might give a good indication about mobility. However, the correlation is not trivial, as there can be travelling with wifi (e.g. train), immobility at places without wifi, and most importantly, a priori it is unknown what portion of the time wifi is available. e.g. sampling heavily during a mobile connection will quickly deplete the battery when this happens for a long time. There will be a correlation between connectivity and travelling, but this is specific to the person and might even change over time.

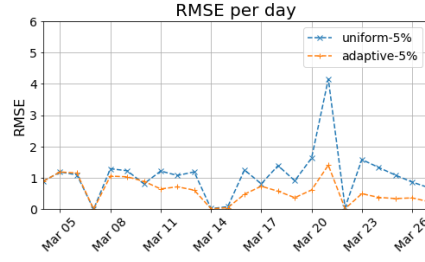
The setup of this experiment is the same as the position experiment with the hour of day as conditional variable, but now the connection type is the condition variable. Instead of a relative frequency of $\frac{1}{24}$ for each hour, the algorithm now keeps online track of the relative frequency of each condition. The three conditions are: no network connection, a network connection over wifi, a mobile network connection.



(a) Adaptive sampling performs approximately twice as well as uniform sampling.



(b) The variation of sample rates learned over time. This is with a 5% sample rate. No connection is very rare (0.2%). As can be seen, the strategy is still adapting to exploit the strong correlation between connection type and travelling.



(c) RMSE over time for adaptive and uniform sampling at 5% sample rate. There is a strong learning trend, with adaptive sampling consistently outperforming uniform sampling over time. It successfully mitigates the peak at March 21.

Figure 6.5: Adaptive sampling on my position data set connection type as conditional variable.

Two observations about the results. First as Figures 6.5c and 6.5b show, the adaptive algorithm is still learning the connection, and over a longer period of time is likely to perform even better. Second, the total number of measurements are as intended (Figure 6.5a), but the number of measurements might vary per day, as connectivity is likely to vary. Figure 6.6 shows the number of samples per day. If there would be an energy budget per day (i.e. daily charging) the adaptive sampling strategy would deplete the battery on a few days. Note that this has not been a problem with the hour of day condition, as the relative frequencies are fixed.

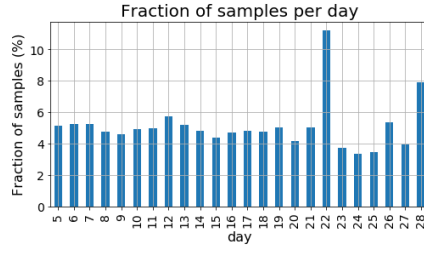
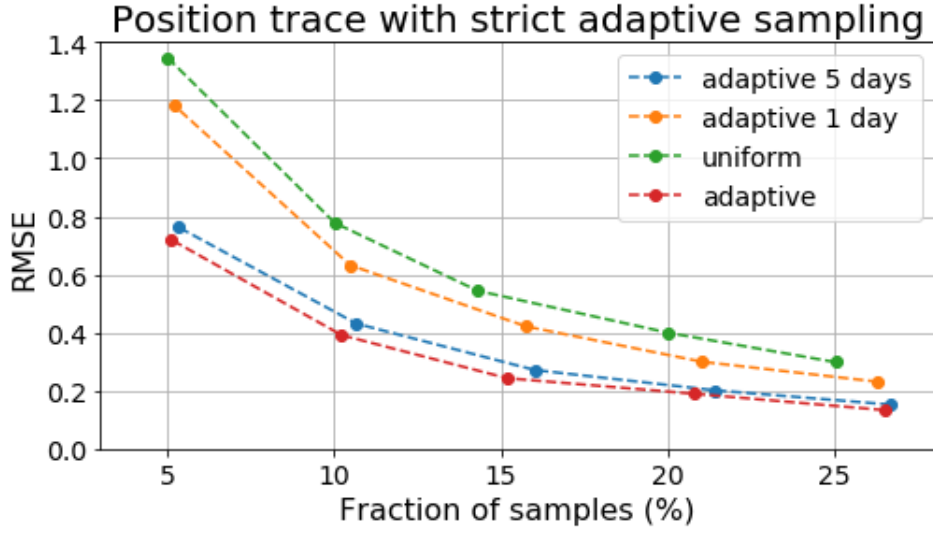
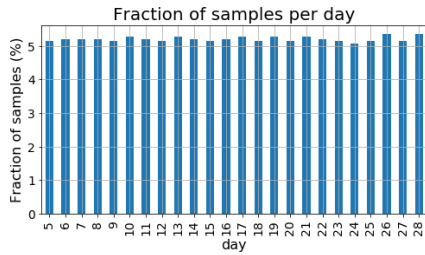


Figure 6.6: The number of samples per day for adaptive sampling at 5%. The number varies a little from day to day and contains a few significant discrepancies.

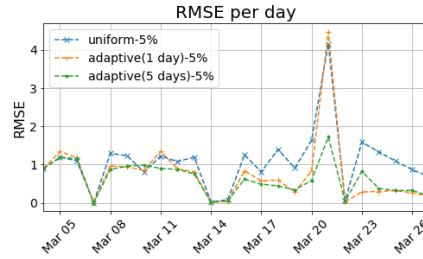
Another experiment was performed with a strict number of samples per day and per 5 days. (as would be the case when recharging every day or 5 days). To enforce a 1 day limitation, the method described in Section 4.1.8 was used. The results are shown in Figure 6.7a.



(a) Adaptive sampling limiting the number of samples per day and per 5 days.



(b) For the adaptive (strict) strategy the number of samples per day varies only slightly.



(c) RMSE over time for adaptive (1 day) and uniform sampling at 5% sample rate. There is a strong learning trend, with adaptive sampling consistently outperforming uniform sampling over time. At the peak it performs worse.

Figure 6.7: Adaptive sampling limiting the number of samples per day yields a slight performance improvement.

Enforcing a period of 1 day diminishes most of the gain. With a 5 day period the improvement is almost as substantial as without the strict enforcement.

6.4 Robustness

In using the algorithm it is important that the algorithm is robust and not overly sensitive to small changes in its parameters. If it is too sensitive it is virtually impossible to use. Another important factor is that a robust algorithm is likely to generalise well to other datasets. To test sensitivity

of parameter t_{half} the experiment on the RTA dataset with hour of day as conditional variable has been repeated with t_{half} ranging from 1 day to 3 weeks. This dataset was chosen because of the long duration over which data is available. Due to the long duration, the initialization time is relatively short compared to the overall duration of the dataset, this reduces the effect of initialization on the experiment. Figure 6.8 shows that the algorithm, at least for the RTA dataset, is quite insensitive to t_{half} .

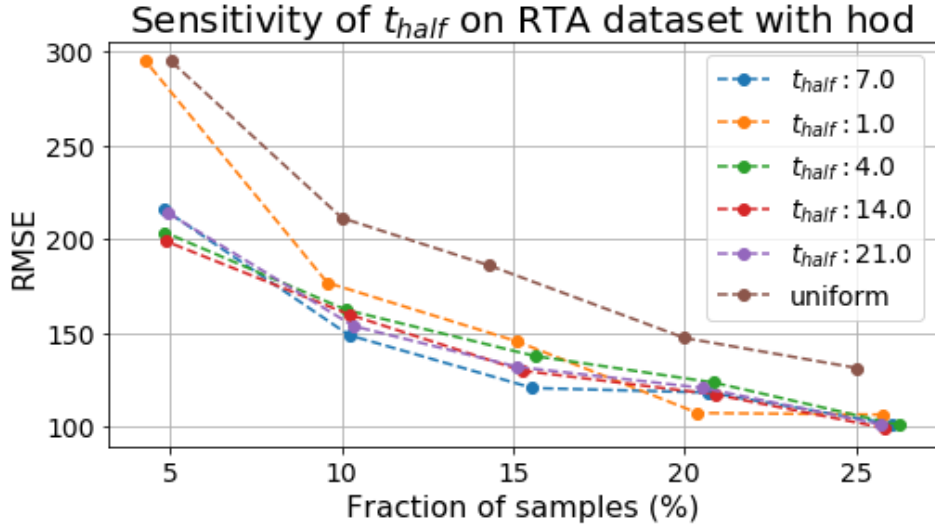


Figure 6.8: Even with t_{half} varying a lot, the RMSE is not affected much.

6.5 Optimality

A lower-bound for the principle of sampling with a uniform prediction error, the principle of the proposed sampling strategy, can be determined. This is done by simulating with an oracle strategy. This oracle strategy ensures that for each sample a sample interval is chosen such that the prediction error for each sample is some fixed value. To do this, the oracle strategy looks into the future, it knows the future signal in order to determine the sample interval.

Even with a uniform prediction error, this does not guarantee that the number of samples is within the energy constraint. Nonetheless, to get an insight, the oracle strategy is simply simulated for different fixed values for the prediction error. Note that this is a very optimistic lower-bound, set by knowing the data in advance. Still, this gives some insight into

- whether the proposed principle of sampling with uniform prediction errors could lead to a big improvement for sampling. And

- whether the proposed sampling strategy is exploiting the principle optimally, or whether there could be much room for improvement.

Figure 6.9 shows the lower bound determined using an oracle strategy. The oracle strategy reduces the RMSE to approximately 50%, whereas the adaptive strategy reaches about 75%. Hence, the adaptive sampling strategy realises approximately half the improvements that can potentially be realised with the principle of sampling with a uniform prediction error.

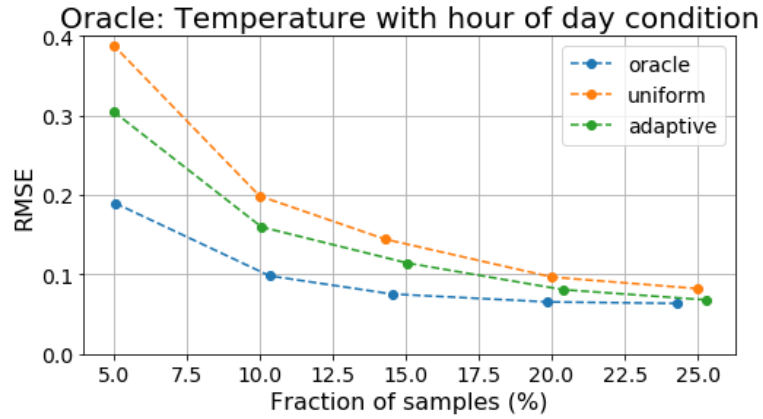


Figure 6.9: The oracle strategy gives a lower-bound for the performance of the principle of *uniform prediction errors*. The adaptive strategy falls somewhere between the uniform and the oracle strategy, showing there is probably some room to improve the strategy.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

An adaptive sample strategy is proposed that specifically addresses the problem of sampling with a strict energy constraint, provided by a time period and a number of samples that have to be taken during that time period. Furthermore, this strategy is signal agnostic in the sense that it doesn't rely on a priori information about the signal but rather learns online how to vary the sample rate from the signal itself. Variation in predictability in a signal is exploited with a controller that varies the sample rate to aim for uniform prediction errors. This strategy has been demonstrated to work with a condition that captures the hour of day, a state for which the probabilities are known ($\frac{1}{24}$ for each hour) and also with a condition for which the probabilities are unknown: the connection type of a smartphone (WiFi, mobile, none).

Simulations on an artificial dataset showed that the system can learn characteristics of a signal and adapt the sample rate to exploit these characteristic. It has also been shown that the system adapts when the signal characteristics change.

Simulations were performed with three real-world datasets: traffic flow (RTA); temperature readings of a wireless node; and a position trace of a person (captured with a smartphone). Using the hour of day as condition variable, reductions in the RMSE, compared to uniform sampling, were shown for the RTA and temperature datasets to be 33% and up to 25% respectively. For the position dataset the the hour of day condition was also exploited successfully, but a change in commute nullified the gains, resulting in a similar performance as uniform sampling. Using another condition, the connection type of the smartphone, the RMSE was reduced by 15% for a recharge once every day, and by 50% for a recharge once every 5 days.

The proposed sample strategy is robust with respect to its only parameter, t_{half} that determines how fast the strategy learns. The performance on the RTA dataset was very similar with t_{half} ranging between 4 and 21 days.

In conclusion, the proposed adaptive sample strategy can be used for sampling a wide range of signals. Based on the results in this thesis, even with a simple predictor, a reduction in the reconstruction error of the signal can be expected. This reduction ranged from 0% (similar performance as uniform sampling) to a reduction by 50%.

7.2 Future work

The experiment with an oracle (see Section 6.5), indicates that the *principle of uniform prediction error* can be exploited more effectively than done so far using the adaptive sampling strategy and an hour of day condition. More specifically, the adaptive sampling strategy, using an hour of day condition, realised half the reduction of the RMSE that the oracle achieved. Although the oracle is based upon an ideal assumption of perfect predictability of the prediction error, other conditions and changes to the current strategy should be tried to narrow this gap.

More suggestions for future work are:

- more advanced predictors; perhaps the results might be improved with more advanced predictors such as Holt-Winters. The used predictor, the signal does not change after a sample, is very basic.
- investigate a burst condition; a condition that states whether the signal is currently in a burst or not (e.g. mobile or immobile) could work very well when used on bursty signals such as human behaviour. Probably the biggest challenge here is that the relative frequency of a burst might vary quite a bit. However this challenge was successfully mitigated for the connection type, the same approach might work well in this case as well.
- continuous conditions; right now discrete conditions are used. But some conditions, like time of day could be expressed as a continuous variable instead of a discrete variable. Modelling time of day as a continuous variable might capture the fact that 2 am and 3 am are related and thus need less time to learn than modelling it as two independent discrete events.
- combine conditions; conditions such as hour of day and connection type or day of the week might be combined. However combining them naively into a their products ((1pm, wifi), (1pm, mobile), etc.) leads to the curse of dimensionality, and thus, might lead to too many condi-

tions with too few samples per condition to learn anything significant over each condition.

Bibliography

- [1] N. Eagle and A. (Sandy) Pentland, “Reality mining: sensing complex social systems,” *Personal Ubiquitous Comput.*, vol. 10, pp. 255–268, March 2006.
- [2] B. Buchli, F. Sutton, J. Beutel, and L. Thiele, “Dynamic power management for long-term energy neutral operation of solar energy harvesting systems,” in *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*. ACM, 2014, pp. 31–45.
- [3] W. K. Seah, Z. A. Eu, and H.-P. Tan, “Wireless sensor networks powered by ambient energy harvesting (wsn-heap)-survey and challenges,” in *Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology, 2009. Wireless VITAE 2009. 1st International Conference on*. Ieee, 2009, pp. 1–5.
- [4] C. E. Shannon, “Communication in the presence of noise,” *Proc. Institute of Radio Engineers*, vol. 37, no. 1, pp. 10–21, 1949.
- [5] J. Paek, J. Kim, and R. Govindan, “Energy-efficient rate-adaptive gps-based positioning for smartphones,” in *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010, pp. 299–314.
- [6] I. Constandache, S. Gaonkar, M. Sayler, R. Choudhury, and L. Cox, “Enloc: Energy-efficient localization for mobile phones,” in *INFOCOM 2009, IEEE*, april 2009, pp. 2716 –2720.
- [7] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell, “The jigsaw continuous sensing engine for mobile phone applications,” in *Proceedings of the 8th ACM conference on embedded networked sensor systems*. ACM, 2010, pp. 71–84.
- [8] C. Alippi, G. Anastasi, M. Di Francesco, and M. Roveri, “An adaptive sampling algorithm for effective energy management in wireless sensor networks with energy-hungry sensors,” *IEEE Transactions on Instrumentation and Measurement*, vol. 59, no. 2, pp. 335–344, 2010.

- [9] Jupyter. (2017) Jupyter notebook. [Online]. Available: <https://jupyter.org>
- [10] C. G. S. M. M. P. Peter Bodik, Wei Hong and R. Thibaux. Intel lab data.
- [11] Kaggle. (2011) Rta freeway travel time prediction. [Online]. Available: <https://www.kaggle.com/c/RTA/data>
- [12] N. Eagle and A. Pentland, “Eigenbehaviors: identifying structure in routine,” *Behavioral Ecology and Sociobiology*, vol. 63, pp. 1057–1066, 2009.
- [13] T. O. Oshin, S. Poslad, and A. Ma, “Improving the energy-efficiency of gps based location sensing smartphone applications,” in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*. IEEE, 2012, pp. 1698–1705.
- [14] P. Mohan, V. N. Padmanabhan, and R. Ramjee, “Nericell: rich monitoring of road and traffic conditions using mobile smartphones,” in *Proceedings of the 6th ACM conference on Embedded network sensor systems*. ACM, 2008, pp. 323–336.
- [15] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, “Energy consumption in mobile phones: a measurement study and implications for network applications,” in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*. ACM, 2009, pp. 280–293.
- [16] M. B. Kjærgaard, J. Langdal, T. Godsk, and T. Toftkjær, “Entracked: energy-efficient robust position tracking for mobile devices,” in *Proceedings of the 7th international conference on Mobile systems, applications, and services*. ACM, 2009, pp. 221–234.
- [17] Y. W. Law, S. Chatterjea, J. Jin, T. Hanselmann, and M. Palaniswami, “Energy-efficient data acquisition by adaptive sampling for wireless sensor networks,” in *Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly*. ACM, 2009, pp. 1146–1151.
- [18] Z. Yan, J. Eberle, and K. Aberer, “Optimos: Optimal sensing for mobile sensors,” in *Mobile Data Management (MDM), 2012 IEEE 13th International Conference on*, pp. 105–114.
- [19] E. J. Candes, J. K. Romberg, and T. Tao, “Stable signal recovery from incomplete and inaccurate measurements,” *Communications on pure and applied mathematics*, vol. 59, no. 8, pp. 1207–1223, 2006.

- [20] D. L. Donoho, “Compressed sensing,” *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.
- [21] H. Mamaghanian, N. Khaled, D. Atienza, and P. Vandergheynst, “Compressed sensing for real-time energy-efficient ecg compression on wireless body sensor nodes,” *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 9, pp. 2456–2466, 2011.
- [22] M. F. Duarte, M. A. Davenport, D. Takbar, J. N. Laska, T. Sun, K. F. Kelly, and R. G. Baraniuk, “Single-pixel imaging via compressive sampling,” *IEEE signal processing magazine*, vol. 25, no. 2, pp. 83–91, 2008.
- [23] M. Lustig, D. Donoho, and J. M. Pauly, “Sparse mri: The application of compressed sensing for rapid mr imaging,” *Magnetic resonance in medicine*, vol. 58, no. 6, pp. 1182–1195, 2007.
- [24] P. S. Kalekar, “Time series forecasting using holt-winters exponential smoothing,” *Kanwal Rekhi School of Information Technology*, vol. 4329008, pp. 1–13, 2004.
- [25] B. M. Williams and L. A. Hoel, “Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results,” *Journal of transportation engineering*, vol. 129, no. 6, pp. 664–672, 2003.
- [26] E. T. Jaynes, “On the rationale of maximum-entropy methods,” *Proceedings of the IEEE*, vol. 70, no. 9, pp. 939–952, 1982.
- [27] C. Song, Z. Qu, N. Blumm, and A. Barabási, “Limits of predictability in human mobility,” *Science*, vol. 327, no. 5968, pp. 1018–1021, 2010.
- [28] J. Inman, *Navigation and Nautical Astronomy: For the Use of British Seamen*, 3rd ed., 1835.

Appendix A

RMSE prediction error

Using a step function for the error (the previous sample is used to predict the next sample)

$$error(\tau) = f(t) - f(t - \tau) \quad (\text{A.1})$$

In order to understand the relation between the error and the sampling interval, the following equation was derived.

Considering $f(t)$ a sample of a stochastic process with a defined mean and variation the expected RMSE given sample interval τ is given by.

$$RMSE(\tau) = \sqrt{E[(f(t) - f(t - \tau))^2]} = \sqrt{2} \cdot \sqrt{\sigma^2 - cov(f(t), f(t - \tau))} \quad (\text{A.2})$$

With σ^2 the variance and $cov(f(t), f(t - \tau))$ the autocovariance (the autocovariance divided by the variance equals the autocorrelation). Figure A.1 on page 50 shows that the formula agrees with the RTA data. For very large sample intervals it seems reasonable to expect independence between samples. Thus giving an asymptot at $\sqrt{2} \cdot \sqrt{\sigma^2}$. For anticorrelated data the RMSE can be higher up to $2 \cdot \sqrt{\sigma^2}$. For a perspective, the RMSE is exactly the standard deviation when predicting the mean. For an autocorrelation between 0.5 and 1 the RMSE is lower than the standard deviation.

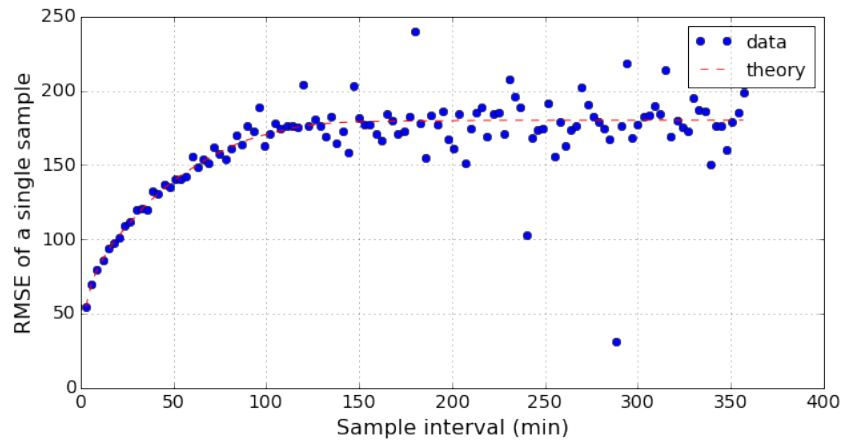


Figure A.1: Root Mean Square Error of samples with a specified interval. The *theory* is the expectation using the variance and autocovariance from the RTA data.