Parsimonious Sensing with Active Learning:
Applications with Context Mining and Environmental Sensing

Intelligent Systems

Manuel Levi Frutuoso

A dissertation presented for the degree of
Master of Science

• U  C •

Faculty of Sciences and Technology
University of Coimbra
September 2015

Advisors:
Bernardete Ribeiro bribeiro@dei.uc.pt
Câmara Pereira camara@mit.edu

Estágio	Parsimonious Sensing with Active Learning: Aplications with Context Mining and Environmental Sensing
Orientador DEI	Bernardete Ribeiro (bribeiro@dei.uc.pt)
Co-Orientador	Francisco Camara Pereira(camara@dei.uc.pt)
Juri Arguente	Alexandre Miguel Pinto (ampinto@dei.uc.pt)
Juri Vogal	Bruno Cabral (bcabral@dei.uc.pt)

Abstract

The unprecedented success of Web 2.0, and with it, social media services, has resulted in massive amounts of user-generated data. Traditional techniques are no longer adequate to deal with this sheer amount of information. In an attempt to address this problem, new techniques that can be applied to big data, are being proposed in an increasingly frequent way.

In this dissertation, the concept of parsimonious sensing and some of its applications are presented. Parsimonious sensing attempts to select the most relevant information from a large dataset, thus reducing the cost of its analysis. To do this, it employs different techniques such as active learning, also know as optimal experimental design in the field of statistics. We also explore some innovative methods of identifying relevant anomalies from a large dataset to be subsequently explored. This dissertation studies the application of parsimonious sensing on three unique datasets. The first main experience studies the employment of active learning in an environmental sensing network system with air quality parameters. The second experience depicts an attempt to predict the number of hits for a certain query related to events happening in Singapore, thus decreasing the number of required queries. The third and last experiment makes use of a dataset provided by a major taxi company in Singapore and tries to identify traffic anomalies and later, synthesize queries that are run through a search engine in order to identify the context of the anomalies.

We found the application of parsimonious sensing to be successful when implemented in the context of environmental sensing. We have further developed a system capable of identifying traffic anomalies and returning a number of links that can potentially explain why they happened. The fully automated system has been shown to be better than a hybrid system, composed of information retrieved both automatically and manually. The findings from this dissertation can hopefully shed some light on the possible applications of parsimonious sensing to diverse contexts.

Keywords. Active Learning, Big Data, Context Sensing, Data Mining, Event Identification, Parsimonious Sensing

Dedication

To my family for caring for my success and to everyone who made this dissertation possible.

Acknowledgements

I have the deepest gratitude for the people who helped me during this journey. Foremost, I would like to express my gratitute to both my adviser and co-adviser, **Bernardete Ribeiro** and **Francisco Câmara Pereira**for all the support shown.I would also like to thank to **Filipe Rodrigues** for all the support given. **Miguel Cabrita** and **Francisco Nibau Antunes** for their encouragement, and help, understanding some of the math necessary for the development of this dissertation.**Penousal Machado** and **Ana Alves**: for trusting me as a researcher in their groups and leading me on diverse projects. Last but not least, I would like to thank my **family and everyone that supported me** during good and bad times, during the course of this work.

Contents

List of Figures

List of Tables

Symbols and Notation

$\triangleq$	an equality which acts as a definition
$\|K\|$	determinant of K matrix
$\|\mathbf{y}\|$	Euclidean length of vector $\mathbf{y}$
$\mathbf{y}^T$	the transpose of vector $\mathbf{y}$
$\propto$	proportional to
$\sim$	distributed according to; example: $x \sim \mathcal{N}(\mu, \sigma^2)$
C	number of classes in a classification problem
$cov(\mathbf{f}_*)$	Gaussian process posterior covariance
D	dimension of input space $\mathcal{X}$
$\mathcal{D}$	data set: $\mathcal{D} = \{(\mathbf{x_i}, \mathbf{y_i}) \| i = 1, ..., n\}$
δ_{pq}	Kronecker delta, $\delta_{pq} = 1$ iff $p = q$ and 0 otherwise
$\mathbb{E}$ or $\mathbb{E}_{q(x)}[z(x)]$	expectation; expectation of $z(x)$ when $x \sim q(x)$
$f(\mathbf{x})$ or $\mathbf{f}$	Gaussian Process (or vector of) latent function values, $\mathbf{f} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))^T$
$\mathbf{f}_*$	Gaussian Process (posterior) prediction (random variable)
$\bar{\mathbf{f}}_*$	Gaussian Process posterior mean
$\mathcal{GP}$	Gaussian Process: $f \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$, the function f is distributed as a Gaussian Process with mean $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$
I or I_n	the identity matrix (of size n)
$k(\mathbf{x}, \mathbf{x}')$	covariance (or kernel) functions evaluated at $\mathbf{x}$ and $\mathbf{x}'$
K or $K(X, X)$	$n \times n$ covariance matrix
K_* or $K(X, X_*)$	$n \times n$ covariance matrix between training and test cases
$K_\mathbf{y}$	covariance matrix for the noisy $\mathbf{y}$ values
$\log(z)$	natural logarithm (base e)
$\log_2(z)$	logarithm to the base 2
ℓ or ℓ_d	characteristic length-scale (for input dimension d)
$m(\mathbf{x})$	the mean function of a Gaussian Process
$\mathcal{N}(\mathbf{x})$	Short for $\mathcal{N}(0, I)$
n	number of training cases
n_*	number of test cases
N	dimension of feature space
$y\|x$	conditional random variable y given x
σ_f^2	variance of the noise-free signal
σ_n^2	noise variance
θ	vector of hyperparameters (parameters of the covariance functions)
$\mathcal{X}$	input space and also the index set for the stochastic process
X	$D \times n$ matrix of the training inputs $\{\mathbf{x}_i\}_{i=1}^n$; the design matrix
X_*	matrix of test inputs
$\mathbf{x}_i$	the ith training input
$\mathbf{y}_i$	the $\mathbf{x}i$th training output

Chapter 1

Introduction

Mankind has been recording its history since the very beginning of writing. Almost 6,000 years have passed now and the world has never recorded so much information as it does right now. For the developed world this means more available knowledge to the general population, but also the problem of information overload, the inability to confirm the veracity of all facts, among others. For entities who want to analyze all this information, such as companies, governments, and academics, this is an excellent opportunity for development, but one highly complex, thanks to the sheer amount of information. In 2010, Google CEO Eric Schmimdt estimated that, every two days, we are creating as much information as we did from the dawn of mankind to 2003. Online social networks and the concepts of OpenData and quantified self push the boundaries on digital storage, communication and computational capabilities. Besides government and individuals, companies, from small to multi-nationals, have also been recording every information they can about their clients and products. This is even more prevalent in a world where sensors are becoming more and more affordable, and readily available to the majority of people living in any industrialized nation. From this collection of information often result datasets so large that, right now, the costs of processing them cannot keep up with their production, these datasets have been called *Big Data*. Many times these datasets can be excessively complex, verbose, or contain a lot of redundant information. These problems are even worse when the information does not follow a specific structure; this is especially the case when the dataset is composed mainly of documents written in natural language, either sourced from a social network or a website such as Wikipedia[1]. Even though the field of Natural Language Processing has been improving each day, the extraction of structured information from this kind of document continues to be extremely challenging. Many times, unless we are looking for very specific information from a dataset, we extract meaning from it by using different sampling methods, the problem with this is that conflicting data is very likely to occur, and the same is for redundant data. The machine learning community is attempting to solve these problems by applying what is known as optimal experimental design in the field of statistics, to the current datasets, resulting in what is now called *Active Learning*. This can be particularly interesting in situations in which raw data is available in large quantities but its labeling expensive, ultimately resulting on the reduction of costs of experimentation and allowing models to be learned with fewer instances. This dissertation starts by exploring the state of the art in regression algorithms,

[1] `http://wikipedia.org`.

active learning and information retrieval from the web (see Chapter 2). Chapter 3 explores the application of an active learning methodology to an environmental sensing problem, where it has shown promising results. Then, Chapter 4 shows an attempt to develop a predictive system, capable of estimating the number of results from a search engine, utilizing techniques such as topic modeling. Our final experiment, described on Chapter 5, explores both the identification of anomalies in a large spatio-temporal dataset and the automatic search for possible explanations.

Chapter 2

State of the Art

In this chapter we will explore superficially some of the techniques used in this thesis.

The first one to be studied is Gaussian Processes, a technique that can be used for regression analysis. This will enable us to make predictions on the air quality for the environmental sensing dataset.

The next section focuses on Active Learning, its application in essential for the thesis and motivates the use of Gaussian Processes against using another regression analysis technique.

2.1 Gaussian Processes

Regression analysis is a very common statistical technique that enables the study of the relationship between variables. First described in the nineteenth century by both Legendre and Gauss, it solved linear problems in theoretical astronomy. Because regression analysis allows the prediction of output values for different inputs, it can help with problems where access to different outputs is expensive, by making guesses based on the previously read data. Unfortunately, without applying transformations to the input variables, linear regressions can not handle non-linear, chaotic patterns, commonly found in real world problems.

This section presents a superficial look on Gaussian Processes (GP). Gaussian Processes are defined by a mean function and a co-variance function, by using a non-linear co-variance function[1], Gaussian Processes can be an approach to non-linear regressions.

2.1.1 Definitions

A real-valued random variable U is said to be a standard normal random variable if it has a probability density function f_U given by

$$f_U(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \ , \ \forall x \in \mathbb{R} \, , \tag{2.1}$$

and we write $U \sim \mathcal{N}(0,1)$. X is said to be a normal random variable if $X = \sigma U + m$, with $m, \sigma \in \mathbb{R}$ and we write $X \sim \mathcal{N}(m, \sigma^2)$.

Figure 2.1 depicts the probability density function for a univariate Gaussian distribution.

[1] In other contexts, commonly named kernel

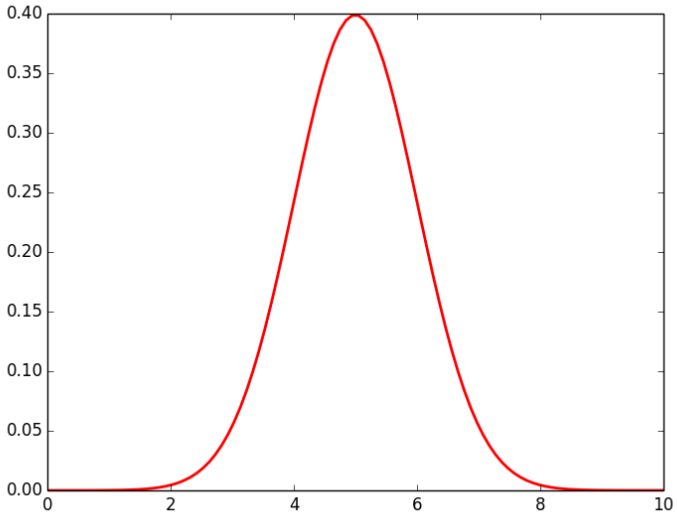


Figure 2.1: Probability Density Function of the univariate $\mathcal{N}(5, 2)$.

The random vector $X = (X_1, X_2, \ldots, X_n)$ is called a multivariate Gaussian vector if $\forall u = (u_1, u_2, \ldots, u_n) \in \mathbb{R}^n$, uX^T is a normal random variable. Let us consider the mean $\mathbb{E}(X) = (\mathbb{E}(X_1), \mathbb{E}(X_2), \ldots, \mathbb{E}(X_n))$ where $\mathbb{E}$ denotes expectation, and Σ the matrix of covariance, with:

$$\Sigma_{ij} = cov(X_i, X_j)\,,\ i, j \in 1, \ldots, n \tag{2.2}$$

and $cov(X_i, X_j)$ is the covariance between X_i and X_j:

$$cov(X_i, X_j) = \mathbb{E}[(X_i - \mathbb{E}[X_i])(X_j - \mathbb{E}[X_j])]\,. \tag{2.3}$$

If the random vector X is Gaussian with mean $\mu = \mathbb{E}(X)$ and covariance matrix $\Sigma = cov(X)$ we write $X \sim \mathcal{N}(\mu, \Sigma)$. The probability density function of X is completely characterized by the parameters $\mu \in \mathbb{R}^n$ and $\Sigma \in \mathbb{R}^{n \times n}$, a symmetric positive-definite matrix.

Figure 2.2 depicts the probability density function for a multi-variate Gaussian distribution.

A stochastic process is a collection $(X_t, t \in T)$ of random variables defined over the same probability space and with values in the same measurable space, where T is the space of the time of the process.

The stochastic process $(X_t, t \in T)$ is said to be a Gaussian Process (GP) with mean μ and covariance matrix Σ if $\forall n \in \mathbb{N}$, $\forall t_1, t_2, \ldots, t_n \in T$, $(X_{t_1}, X_{t_2}, \ldots, X_{t_n})$ is a Gaussian vector over $\mathbb{R}^n$, and we write

$$X \sim \mathcal{N}(\mu, \Sigma). \tag{2.4}$$

A GP is, therefore, completely characterized by μ and Σ. GPs can be thought of being an infinite-dimensional generalization of the multivariate normal distribution making it a set of random variables indexed by a continuous variable: $f(x)$. Because the GP is a conditional probabilistic model, while the probability of f given X, $p(f|X)$ is specified, the distribution on the inputs $p(x)$ is not. It is also worth noticing that GPs however, are not necessarily stochastic processes, where time plays a role.

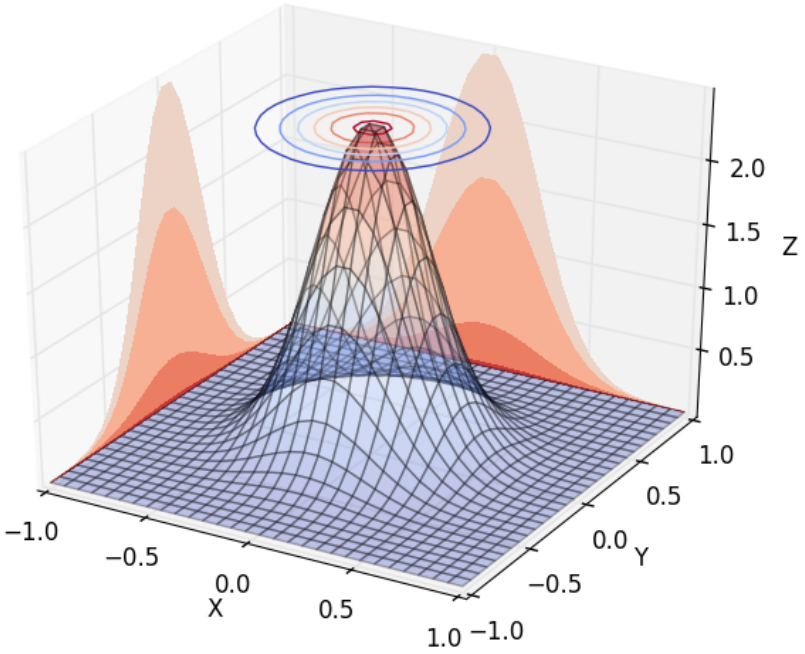


Figure 2.2: Probability Density Function of (X_1, X_2), where both variables have Gaussian distribution $\mathcal{N}(0, 0.25)$.

2.1.2 Covariance Functions

To specify a particular GP prior, the mean μ and covariance matrix Σ of 2.4 need to be defined. It is very common to use a zero mean for the prior along with a zero mean normalization of the dataset such as the z-score, using the formula:

$$z_{score} = \frac{X - \mu}{\sigma} \tag{2.5}$$

We shall set the prior mean to zero, but the posterior GP $p(f|\mathcal{X})$ will, most likely, not have a zero mean process. If a GP is assumed to have zero mean, the choice of the covariance function K to construct the covariance matrix Σ completely defines the process' behavior. Writing 2.3 in vector form, we have:

$$K(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(\mathbf{x} - \mathbb{E}[\mathbf{x}])(\mathbf{x}' - \mathbb{E}[\mathbf{x}'])^T] \tag{2.6}$$

for different points, $\mathbf{x}$ and $\mathbf{x}'$. This covariance function effectively defines how similar two instances are. By utilizing a non-linear covariance function, we can map a given space into some other (usually very high dimensional) space, avoiding the explicit mapping that is needed to get linear learning algorithms to learn non-linear functions or decision boundaries, in the case of classification.

Provided that the covariance matrices produced are always symmetric and positive semidefinite we can choose any covariance function we want. For the best results, a GP should use an appropriately selected covariance. Although there is

a variety of covariance functions available to be used, Squared Exponential (SE) continues to be the most popular when explaining Gaussian processes [7, 53]. This function, also known as Radial Basis function kernel, provides very smooth sample functions, infinitely differentiable. It is also stationary[2] and isotropic[3] and it is defined by:

$$cov(f(\mathbf{x}_i), f(\mathbf{x}_j)) = k(\mathbf{x}_i, \mathbf{x}_j) = a^2 exp(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\lambda^2}), \tag{2.7}$$

where changing the *hyperparameter* a changes the amplitude of the function, *lambda* is the characteristic length-scale[4] of the system and $||\mathbf{x}_i - \mathbf{x}_j||$ the euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_j$.

Related work shows the SE to be too smooth to model natural data [27, 49] and that it can be useful to experiment with different kernel functions. Refer to Appendix A to a comprehensive list of kernel functions, including definitions and samples.

2.1.3 Regression

Assuming noise-free observations, to perform regression using Gaussian processes we use the following formula:

$$f_*|X_*, X, f \sim \mathcal{N}(K(X_*, X)K(X, X)^{-1}f, \tag{2.8}$$
$$K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*))). \tag{2.9}$$

It is, however, too optimistic to not account for instrumental noise and other measuring problems in the real world. In order to perform regression with noisy observations, we assume y to be $f(x) + \epsilon$ and ϵ to be independent identically distributed Gaussian noise with covariance $\sigma_n^2 I$. Since for any random D-dimensional variables X and Y, $cov(X + Y) = cov(X) + cov(Y) + cov(X, Y) + cov(Y, X)$, we have

$$cov(y) = K(X, X) + \sigma_n^2 I, \tag{2.10}$$

because $cov(f(x), \epsilon) = cov(\epsilon, f(x)) = 0$ since ϵ is independent from $f(x)$, ultimately resulting in:

$$f_*|X, y, X_* \sim \mathcal{N}(\bar{f}_*, cov(f_*)), \text{ where} \tag{2.11}$$
$$\bar{f}_* \triangleq \mathbb{E}[f_*|X, y, X_*] = K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1}y, \tag{2.12}$$
$$cov(f_*) = K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1}K(X, X_*). \tag{2.13}$$

The result of this application can be seen in figure 2.3.

[2]The result of a stationary covariance function is defined by the separation $x - x'$ and not on the actual position of x and x' making it invariant to translations

[3]The result of a isotropic covariance functions is invariable to both translation and rotation

[4]Set for each input dimension, the characteristic length-scale defines "how far apart" two points $\mathbf{x}$ and $\mathbf{x}'$ have to be for X to change significantly. The characteristic length-scale is also known as *bandwidth parameters*

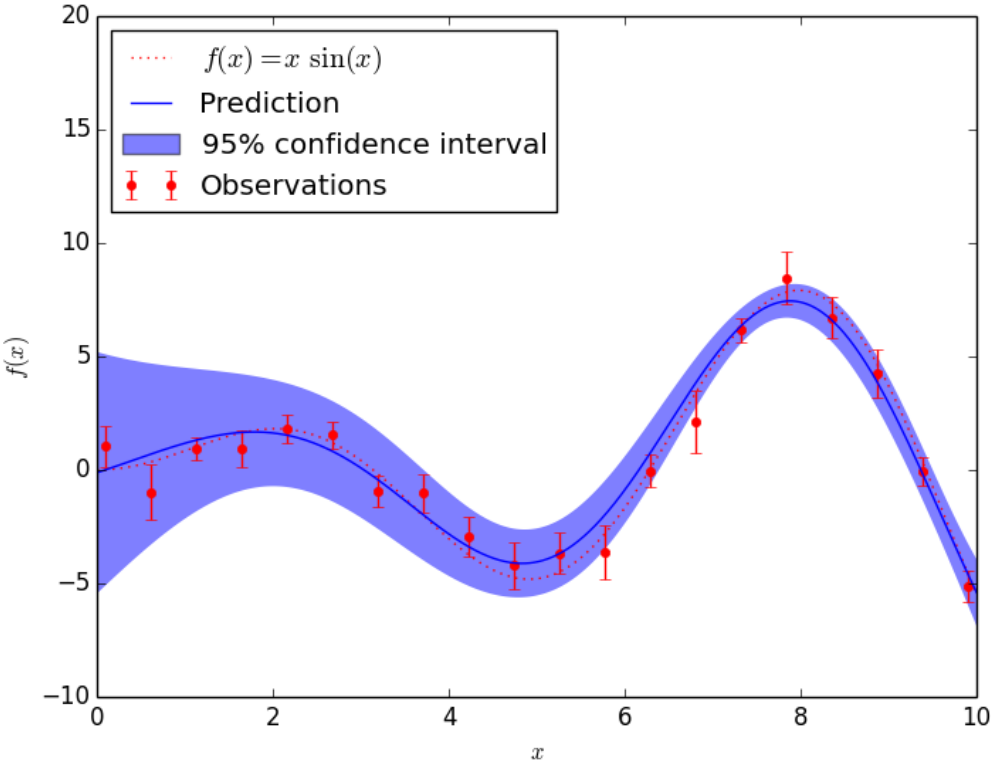


Figure 2.3: GP regression.

2.2 Active Learning

Active learning, also called optimal experimental design, aims to maximize a model's performance while minimizing the number of necessary labeled instances. A hypothesis is that given the algorithm's power to choose which points to learn, it will be able to accomplish the same performance with fewer data, making it computationally more efficient while requiring fewer training data. In the context of this thesis, the selection of relevant data is one of the critical success factors if we want to reduce the cost of data collection.

Active Learning can be a powerful technique when applied to a dataset that, regardless of its size, contains a high percentage of unlabeled instances and these instances are expensive to label. There are a range of problems sharing this characteristic, one is in the field of Evolutionary Computation [50], where there is extensive work on reducing it, especially when a human is necessary to subjectively rank the result of the system: to answer an unformulable question (e.g. "How beautiful is this picture?"); another example of this is mine prospecting, when each exploration for analysis costs a company time and money.

With the recent growth in datasets' size came, the never so popular, big data. Some people have described big data as information so large and complex that its mining process can be very difficult, nearly impossible, to achieve through traditional methods. The application of active learning has been proposed to deal with this kind of problems before as it makes an effort to select the most informative or representative instances to the detriment of others considered redundant [22]. This

dissertation attempts to show the successful application of active learning to the modeling of a large dataset.

2.2.1 Active Learning Settings

There are three main active learning settings in the literature. Two of them are described superficially while the last one is described in greater detail as this is the one used in the dissertation.

Membership Query Synthesis The system can ask for any instance possible to be generated in the search space. This can lead to unexpected problems as it can be hard to limit the search space so that only semantically correct instances are created [4].

Stream-based Selective Sampling or *Sequential active learning* makes the assumption that getting unlabeled instances is free or inexpensive enough so unlabeled instances can be sampled and then decided if asked to be labeled or not.

Pool-based Sampling The pool-based active learning cycle can be explained visually by figure 2.4. It assumes that a large number of unlabeled instances can be gathered at once with a small set of labeled data. In this setting, a model starts with a pool of unlabeled instances from which it selects one or more instances to be labeled by an *oracle* [5]. Then using the resulting training data (i.e. labeled data returned by the oracle) the system creates a new model of the data. In the next step, the system selects a new instance or instances to be labeled and added to the training data, completing the cycle.

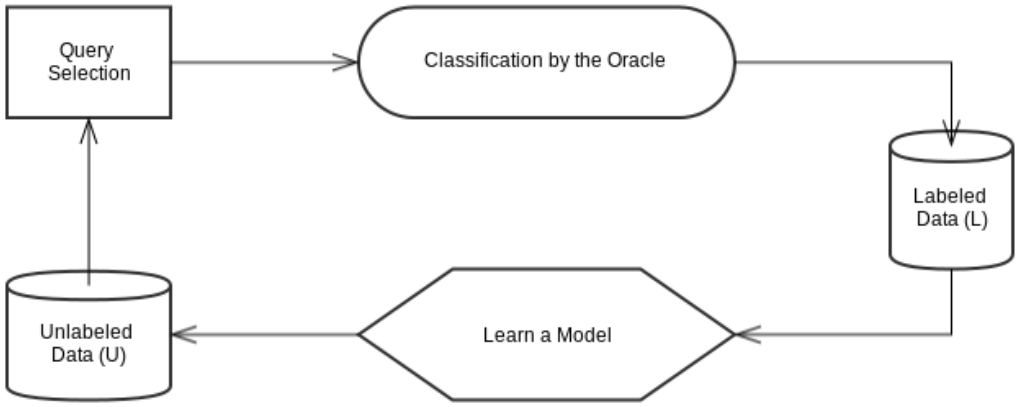


Figure 2.4: Pool-based sampling.

2.2.2 Query Strategies

There are at least six main query strategies in active learning [43]:

[5] An element of the system capable of labeling instances.

Uncertainty Sampling [30]: the system queries the instances which it is least certain about;

Query By Committee [47]: the system maintains a committee composed by various models, each model then votes on the instances label, finally the instance with most disagreement on predicted labels is queried. This has been used successfully with regression problems too [9];

Expected Model Change [46]: the system selects the instance that would lead to the greatest change to the current model if its label is known;

Expected Error Reduction [36]: as the name implies the system queries the instance more likely to reduce the error of the model;

Variance Reduction [18]: the system minimizes the expectation of a loss function *indirectly* by minimizing output variance;

Density Weighted Methods [44]: the system not only looks for instances it is uncertain about, but also the ones that are similar to many others and, that way, "representative" of the underlying distributions.

2.2.3 Uncertainty Sampling

Since Gaussian processes formally incorporate uncertainty, the most natural strategy to use is uncertainty sampling.

Its use is straightforward for probabilistic learning models. For the classification problem, it might be as simple as querying the instance the classifier is the *least confident* about:

$$x^*_{LC} = \arg\max_x 1 - P_\Theta(y'|x), \tag{2.14}$$

where $y' = \arg\max_y P_\Theta(y|x)$, the label with the highest posterior probability according to model Θ. This strategy was used before, for example, in information extraction tasks for the development of statistical sequence models [12, 31, 45].

Margin sampling [41] or *Best vs Second Best* [25], is also a proposed alternative. Instead of considering only information about the most probable label, it also considers the second most probably label:

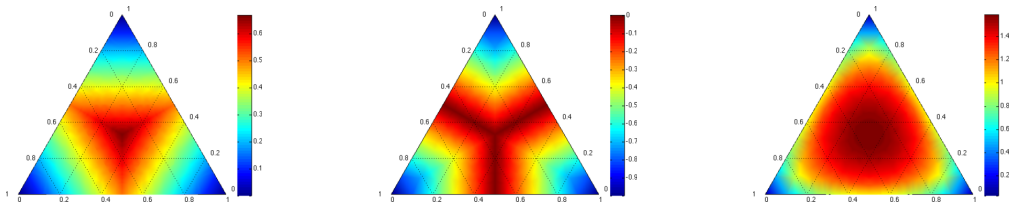
$$x^*_M = \arg\min_x P_\Theta(y'_1|x) - P_\Theta(y'_2|x), \tag{2.15}$$

where y'_1 is the most probably label and y'_2 the second most. It works under the assumption that instances with small margins are more ambiguous, and knowing their true label would help the model more than it would if just choosing the ones it is least confident about. However, when there are a large number of possible labels, using just the top two most leads the model to ignore much of the remaining classes. Using *entropy* [48] as an uncertainty measure is also a very popular choice:

$$x^*_H = \arg\max_x -\sum_i P_\Theta(y_i|x) \log P_\Theta(y_i|x), \tag{2.16}$$

where i ranges over the number of possible labels. In information theory, entropy is the average amount of information that is conveyed by a message. The idea is that the less likely an event is to occur, the most informative it is when it happens. For binary classification, all the above uncertainty measures do the same thing, they select the one with a class posterior closest to 0.5. However, in some cases, entropy showed better results in problems where there is a large number of labels, such as object recognition [21] and image classification [25], and more complex structures such as sequences [44] and trees [23].

Figure 2.5 illustrates how these different uncertainty measures change the query behavior. In all of the triangles, the most informative instance is located in the center where the posterior label distribution is equal for all labels, while the least informative instances are at the corners, where the classifier is completely certain of their classification.



(a) Least Confident. (b) Margin. (c) Entropy.

Figure 2.5: Different query behavior in a three-label classification problem [43].

It is worth noticing that because entropy is influenced by the probability values of unimportant classes, it can also have its drawbacks in comparison with margin and least confident sampling. For example, entropy does not favor instances where the classifier is confused only about two classes. Figure 2.6 illustrates an example in which the classifier is very confused about which of two different labels should be assigned; while in the second, it is much more confident about one specific label, even though the entropy measure is higher.

Multiple authors have compared these different uncertainty measures, producing mixing results [25,28,42,44]. This suggests that the choice of the uncertainty measure is application dependent, much like the *no free lunch theorem* for classifier problems.

2.3 Information Retrieval and Event Identification from the Web

Search engines use crawlers to explore the web, opening link after link and saving everything, from text to images, from hyperlinks to mobile capability from every page visited. The creation of Google and other search engines, such as Bing and Yahoo! changed the way how most people look for information. With all this information available at our fingertips, scientists are trying to use the application of search engines in less traditional ways. People are now open to the idea of using the Web as a Corpus for linguistic research. It has already been used for the most different analysis, from specific tasks such as finding n-gram frequencies [35] to detection of plagiarism [51] and even greater tasks such as creating a Catalan corpus from the Web [26].

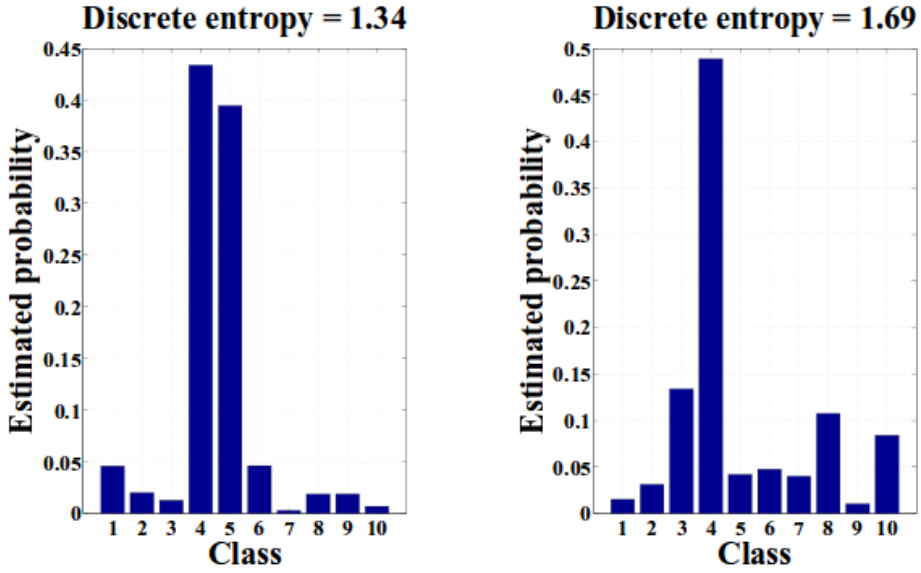


Figure 2.6: In a 10 class problem, entropy can be a poor estimate of uncertainty [25].

2.3.1 Query limitations

One must keep in mind its unique characteristics while using a search engine for information retrieval. While people are trying to make search engines more user-friendly, this results in some added complexity and, sometimes, limitations for the user.

One example of this is when Google and other search engines try to make use of context expecting to provide the user better quality results. An example of this can be observed when searching for *health*, *care* and *health care*. While searching for *care* only, most of the results are pages where the word is used as a verb, while in *health care* most of the results would show the usage of care as a noun. Worse yet, the search engine can also search for the queries together as one single entity, even in cases where it is just by chance that these words got in this specific order. Furthermore, many search engines remove punctuation which could result in the incorrect junction of these two words as just one entity, when, in fact, they are part of two different sentences [35].

All these features can cause problems while automatically generating queries to search on any search engine.

2.3.2 Query Expansion

Intelligent search engines such as Google, Bing or Yahoo! tend to find synonyms of words, different morphological forms of words, or different date formats. This is called query expansion.

In some cases, this can be advantageous to the user while in others it can return unusual results, especially when a day is converted to a time of the day. In this case there can be incorrect conversions between date formats, e.g. *12-08-2012* can be converted to a query that looks for a specific time such as *12:08* during 2012. This can be especially problematic if these results are shown in the first ten documents or usually, the first page of hits, since 62% of users do not look past these [24].

2.3.3 Number of Results

Search engines are nowadays too complex to run on a single machine and requests are served by multiple servers. Not only that, but search engines need to be constantly updating themselves for new information or removal of websites that no longer exist. Even the indexes for the databases are spread across a distributed system [8]. This creates a problem known as search engine "dancing": if two machines are queried at the exact same time and are not synchronized between each other, the number of search results will vary. Furthermore, most of these search engines do not return the exact number of hits, but only an estimate, unless the number is relatively small, which can be a problem while performing analysis over these results.

2.3.4 Event Identification

Social media sites, such as Twitter, Facebook and Youtube are just some of the websites where people choose to share their life experiences to the world. From birthday parties to concerts by a popular music band, there is a wide variety and range of event types that can be found using these websites. News on these websites can spread even prior to the traditional news media [29, 37]. In a paper that analysis a stream of tweets from Europe during a European Football Championship, the authors compare users that post messages close to an event to human sensors, capable of describing an event [1].

The creation of this systems can be useful for different entities, such as a goverment, interested in keeping public safety with crowd control, and traffic management; event organizers, so the can benchmark the effect of their marketing campaigns used to attract people or the origin of their public [11].

Most of these efforts have focused on using social media in general, and Twitter in particular [5, 37, 40, 52]. Twitter has the advantage of the possibility to access the data as a stream, receiving information as it is produced and this has been applied in systems capable of identifying events happening in real time.

However, using Twitter and other social media as sources has some challenges associated, one is the immense scale of data, while the other is the heterogeneity of information present, which has been divided before in 9 different categories, from self-promotion to questions to followers, none of which related specifically to events [34].

Usually, to cope with this problem these systems cluster related information and then classify these clusters into event related and non-event related groups [6].

A more general approach used a large publicly available web corpus, ClueWeb12, and a set of 217,000 unique events to identify pages that contained information about these events. For this, documents were scored according to the page's likelihood of discussing an event, taking a naive-Bayes approach and using a bag of words, assuming that all terms are independent, estimating the probability of a whole document by the probability of all its component terms [17].

Even identification has also been proposed using Call Detail Records, are logs of user transactions with a mobile service provider, to identify places where the density of people present is above normal. In a specific case, a system capable of identifying events in the city of Mons in Belgium was used by the police and organiser to monitor the events during the opening ceremony of Mons as the Europen City of Culture in 2015 [11].

Chapter 3

Parsimonious Sensing

When we are retrieving information from the environment were are most likely engaged in some kind of context sensing application. Called key to the future development of smart environments and applications [16], context sensing is defined as the collection, transformation, interpretation, provision and delivery of context information [2,13]. Frequently there is a strong connection between context sensing and ubiquitous computing [20,39]. The most recently added tool in the field of context sensing and ubiquitous computing is, without a doubt, the smartphone. After being accepted by the great majority of the public, smartphones are now packed with features such as: various sensors, unprecedented cpu power, localization capabilities (GPS), good battery lifetime and access to the internet, making them an affordable alternative to more expensive wireless sensors. These applications however are limited to the available sensors present on the phone, and focus mainly on providing context to applications present on the smartphone itself [10,14,38]. For other applications such as air quality monitoring or industrial system's management, a more traditional approach is still the norm. Air quality, health, and ultimately, quality of life, are directly connected. With the rapid growth of global urban populations, the concern for global health is increasing too. The World Health Organization lists air pollution as a major environmental risk to health, as it has a direct relationship to the incidence of diseases such as lung cancer, chronic and acute respiratory diseases (e.g. asthma), heart diseases, and others. Even though the environment remains at the forefront of scientific interest, the need to better understand the patterns and processes that characterize it, continues to exist. Information plays an important role, since it can have a definitive impact on the quality of environmental decisions. This chapter will focus on the problem of collecting that information from the environment, using limited resources. Fortunately, technological developments in recent years brought access to affordable Internet-enabled sensors which motivated the development of new environmental sensing projects worldwide. At least two academic projects measure air quality parameters. That is the case for the Citisense [3], a network of mobile pollution sensors developed by the University of California San Diego and for Clairity[1], a network of air quality sensors spread around the Massachusetts Institute of Technology campus, in Cambridge, Massachusetts.

[1] urlhttp://clairity.mit.edu.

3.1 Dataset

After evaluating different datasets, we finally chose to use Clairity's dataset, for two main reasons: data availability and fixed sensor placement, which has been for years the reality of environmental sensing. The dataset is split across different files, one for each sensor. Each of these files is composed by a time stamp and a column for each air quality variable: CO, NO, NO_2, O_3, *fine particles* and *big particles*.

3.2 Exploratory Analysis and Data Preprocessing

This section explores the various steps of data exploration for this dataset. In it we present a broad study over the dataset variables and different locations, we also attempt to explain different correlations found while doing this analysis.

3.2.1 Air Quality Variables

A first glimpse of the dataset showed unexpected column names. Even though the columns for small and big particles had a simple, direct correspondence with columns named "small particles" and "big particles", we could not establish the same connection for the other concentration levels like CO and NO. Instead, the columns were named "dylos bin x" where x would go from 0 to 4. We could also not find anything about the used labels for each column on the project's documentation or web page.

Before stepping into a further analysis, we wanted to see how problematic this could be. A very high correlation between variables could mean that we would not need to study all the variables to test our models' capabilities.

To confirm that the variables had a strong relationship between each others the following steps were applied to a 1,000 observations sample:

- Normalize each column using the feature scaling formula, see Formula 3.1;
- Plot a multivariate plot, see Figure 3.1;
- Plot their correlations, see Figure 3.2;
- Analyze their correlation, see Table 3.1.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3.1}$$

	big particles	small particles	dylos 1	dylos 2	dylos 3	dylos 4
big particles	1.000	0.868	0.823	0.896	0.952	1.000
small particles	0.868	1.000	0.993	0.960	0.917	0.868
dylos 1	0.823	0.993	1.000	0.925	0.872	0.823
dylos 2	0.896	0.960	0.925	1.000	0.943	0.896
dylos 3	0.952	0.917	0.872	0.943	1.000	0.952
dylos 4	1.000	0.868	0.823	0.896	0.952	1.000

Table 3.1: Correlation matrix between variables.

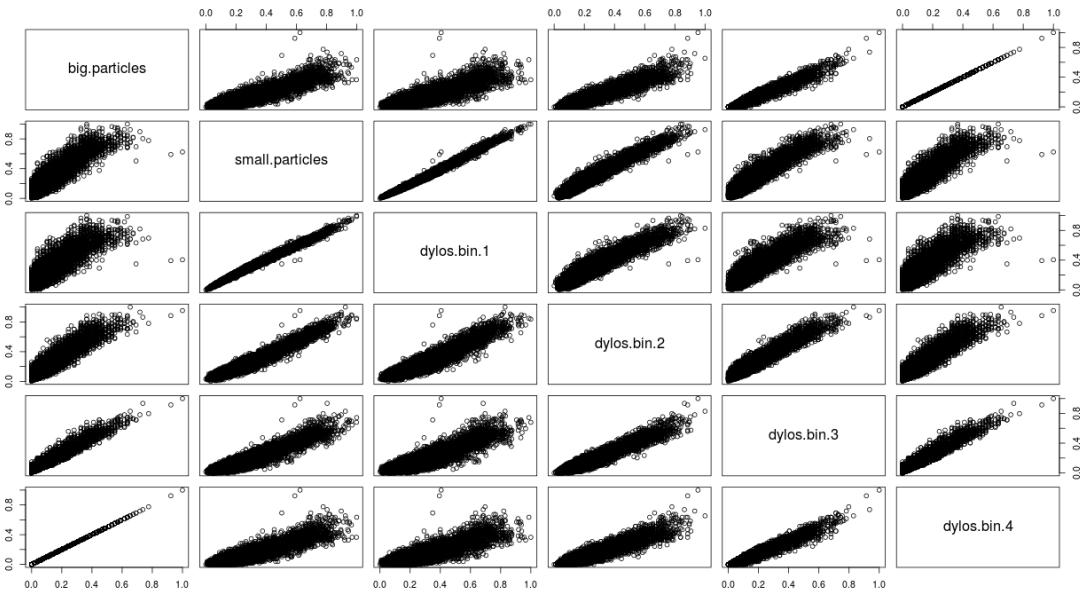


Figure 3.1: Multivariate plot.

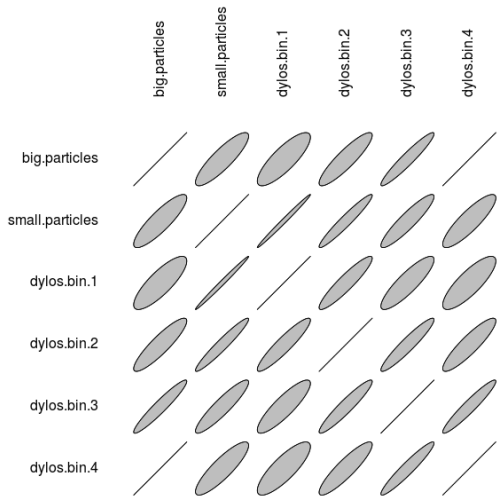


Figure 3.2: Visualization of the correlation matrix between variables.

According to an environmental modeling adviser, Dr. Lynette Cheah, ideally, an environmental study should focus on the CO_2 concentration, but, given there is a mean of 0.92 for all correlations, for the purpose of this analysis, using fine particles would be an appropriate choice to use as an example.

3.2.2 Missing Data and Sampling Times

Having chosen to study the concentration of fine particles, the next step was to plot information from different nodes. From plots like these we could visually conclude that there were some missing values, and high correlation between some nodes. Figure 3.3 shows the number of small Particles per $0.01ft^3$ during the month of May for two different sensors which are highly correlated.

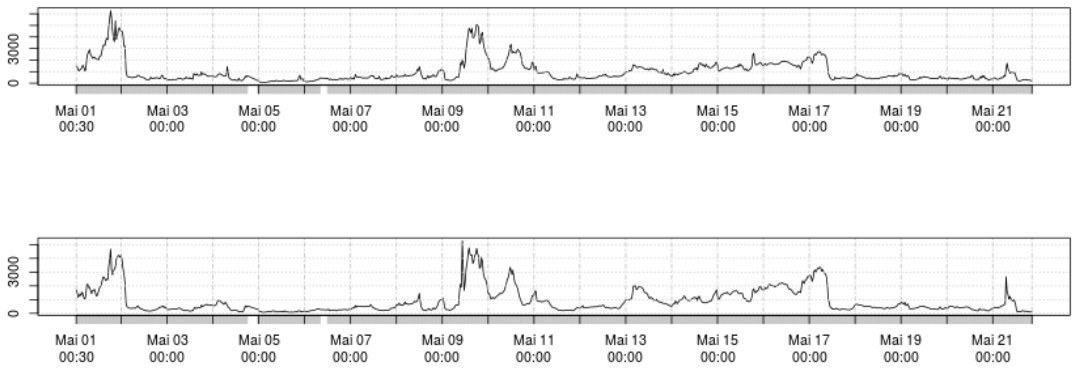


Figure 3.3: Small particles per day, showing two high correlated sensors.

Figure 3.4 shows a sensor that is highly correlated with the ones shown on the previous Figure 3.3, but, because there is absolutely no data after the thirteenth of May, it might not seem so.

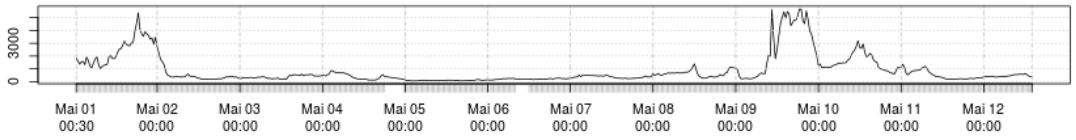


Figure 3.4: Small particles per day, showing a sensor location with incomplete data.

Most of the Clairity project is open-source and can be found online[2]. After a quick analysis of the available code, I realized some sensors had their name changed but not their identifier. If this was the case, then, when downloading the data for a specific place, we could actually be unknowingly downloading data for two different places thinking it was only one, the last to be named.

When contacted about the missing data and the sensor ID's problem, the leader of the project, Eben Cross, confirmed these problems and stated that the network

[2] `clairity.mit.edu`.

was still being evaluated and only data from April to July should be considered valid. Cross also said that the change of the sensor's placement was not public, but everything was recorded and that this information should be released little after this contact. Both of these problems had a negative impact in our analysis, because, since the first records are dated from 24th of April onward, this left us with only three months for our analysis: May, June, and July. To this day, no information has been published on their website.

There was, however, another problem: missing data for the months in which the sensors have not been moved out of place. For some reason, some of these sensors still had no readings at specific times during this period. Furthermore, because this missing data was irregular, it caused the time series to be unevenly spaced. It is, although, very common for time series not to have constant spacing of observation times. This happens in the stock market, geology, climatology and even in clinical trials, where patients can not be observed at regular time intervals. Ideally, this kind of time series are analyzed unchanged, however, most algorithms favor the analysis of evenly spaced time series for various reasons, as linear algebra can be applied in an efficient way in these cases. Because this is a common problem , there are some well-known solutions for the problem of converting an unevenly spaced time series to a regularly spaced time series. One way to do so is by using some form of interpolation but, when the space between observation is highly irregular, which is Clairity's case, the introduction of errors is very likely to occur. Another method tries to solve this, reducing the resolution of the time variable by applying the mean function in regular spaced intervals. Having decided in favor of this, different values for the intervals were evaluated showing a trade-off between the correlation mean between nodes and both the number of incomplete[3] and complete observations[4].

By plotting the evolution of these values, we could identify, the periods with good semantic value and where the information loss is not yet a problem. This can be seen in Figure 3.5 and Table 3.2.

Sampling time in Minutes	Correlations mean between nodes
1	0.701
5	0.719
10	0.733
15	0.739
20	0.745
30	0.750
45	0.764
60	0.767
90	0.771
120	0.775

Table 3.2: Correlation mean and sampling times.

[3]Observations where data exists only for some columns, in this case, for every node.
[4]Observations where data exists for every column.

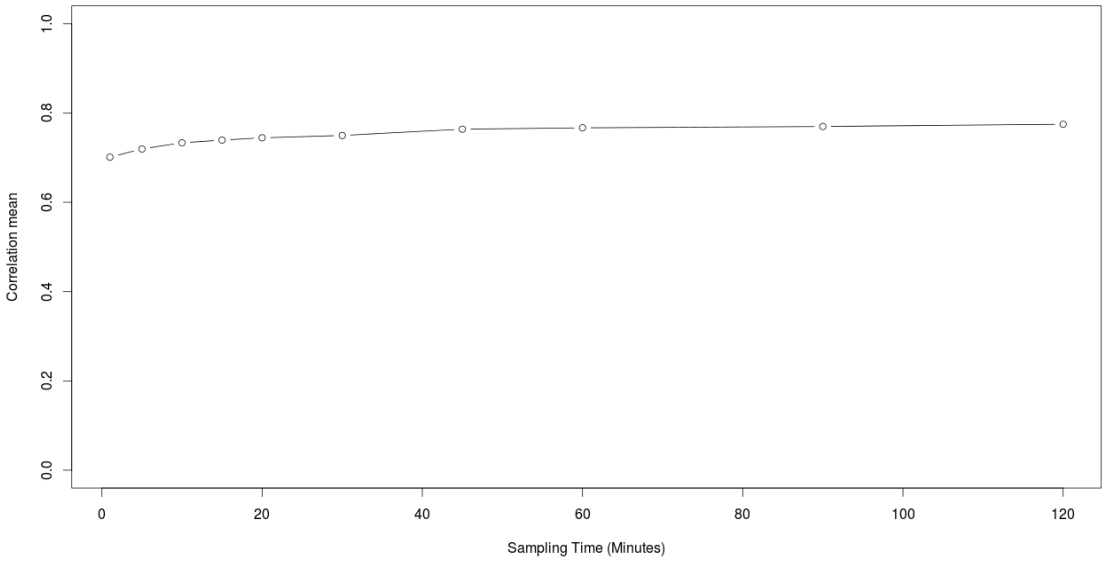


Figure 3.5: Correlation mean and sampling times.

We can see that, as expected, the longer the sampling time was, the more similar nodes were between each others, likely because the longer the sampling times are, the less noisy and sensitive to short changes the data tends to be. Weather conditions also tend to impact outdoor locations immediately, while indoor locations, due to their typical weather proofing tend to be less susceptible. We can also take from this analysis that there is a very strong correlation mean ($r > 0.7$) between all nodes, verifying our suspicions from the preliminary analysis.

Because the correlation mean varies very little with these relatively small sampling times, the next step was to analyze how the number of complete observations is related with the sampling time. Similarly to the *elbow method*[5], we could use Figure 3.6 to help us selecting the right sampling time for our analysis, which values can be seen in Table 3.3.

Sampling Time in Minutes	Complete observations
1	868,606
5	178,558
10	89,476
15	59,727
20	44,854
30	29,965
45	20,027
60	15,048
90	10,089
120	7,592

Table 3.3: Complete observations and sampling times.

[5]Commonly used method for determining a good number of clusters in a dataset by plotting the percentage of variance against the number of clusters and selecting the number of cluster from which the marginal gain drops onward.

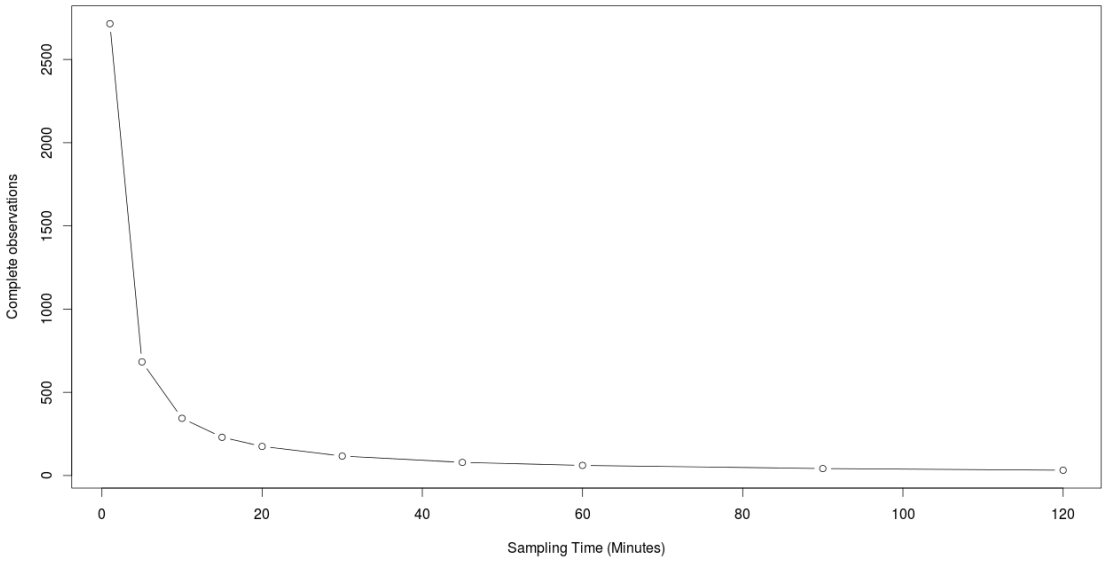


Figure 3.6: Complete observations and sampling times.

A 30 minutes sampling time proved to be the right choice, because:

- Each day would be divided with very little effort into 48 sections, one for each half-hour;
- It is a reasonable resolution for this application;
- It reduced the number of complete observations to 3% of the original which was critical to facilitate future processing; [6]
- It reduced the probability of instrumental error.

With 30 minutes sampling, each node has a range of observations between 139 and 3,875, far from the 5,808 expected for the range between the beginning of April and end of July and 4,656 from May to July. Table 3.4 summarizes this information while Figure 3.7 shows this information visually.

[6]Gaussian Processes complexity is $\mathcal{O}(N^3)$, because of the inversion of an $N \times N$ matrix.

Node	Number of observations (30 min sampling)
Expected (April to July)	5,808
Expected (May to July)	4,656
West_Parking_Garage	3,875
Stratton_Student_Center	3,604
Cafe_4	3,572
Next_House_Courtyard	3,489
MIT_Medical_Parking	3,434
Briggs_Field	3,317
Parsons_Laboratory	2,980
Burton_Conner	2,910
Next_House_Dining	2,652
Green_Building	2,394
Building_16	1,936
Mass_Ave	1,903
Walker_Memorial	1,681
MIT_Museum	1,407
Green_Building_Roof	1,245
Killian_Court	1,167
Stata_Loading_Dock	998
Media_Lab	792
Sloan_School	312
Building_1	139

Table 3.4: Number of observations for each node (30 min sampling).

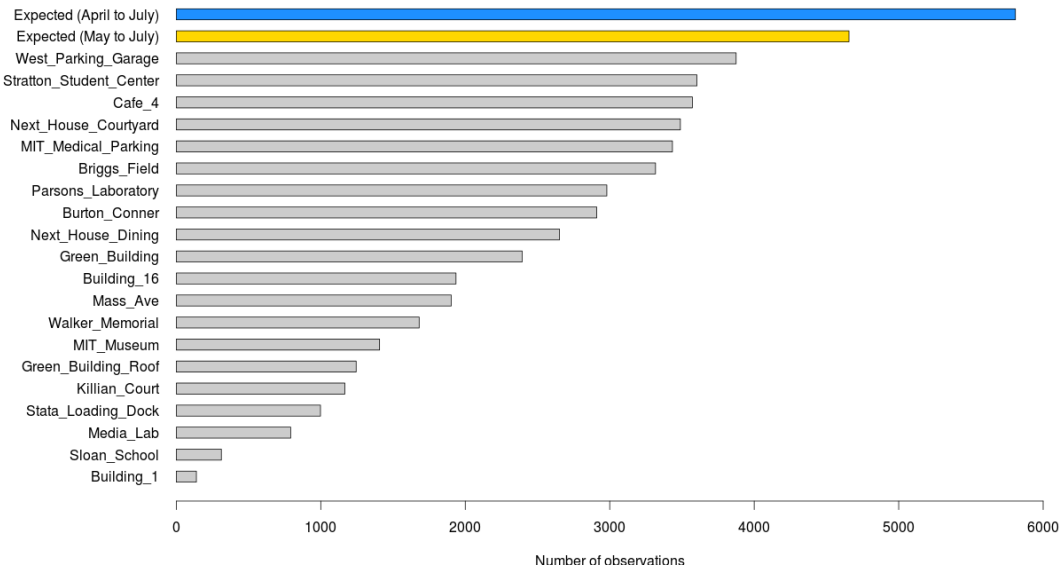


Figure 3.7: Number of observations (30 minutes sampling).

3.2.3 Relationship Between Different Locations

The following step was to compare the values of each sensor location to each other. Again, missing data created some problems, as some analysis did not have enough statistical significance to be considered valid.

By calculating the correlation matrix we can see how much the behavior of one node is similar to another at any given time. A simple visualization of these values can be seen in Figure 3.8.

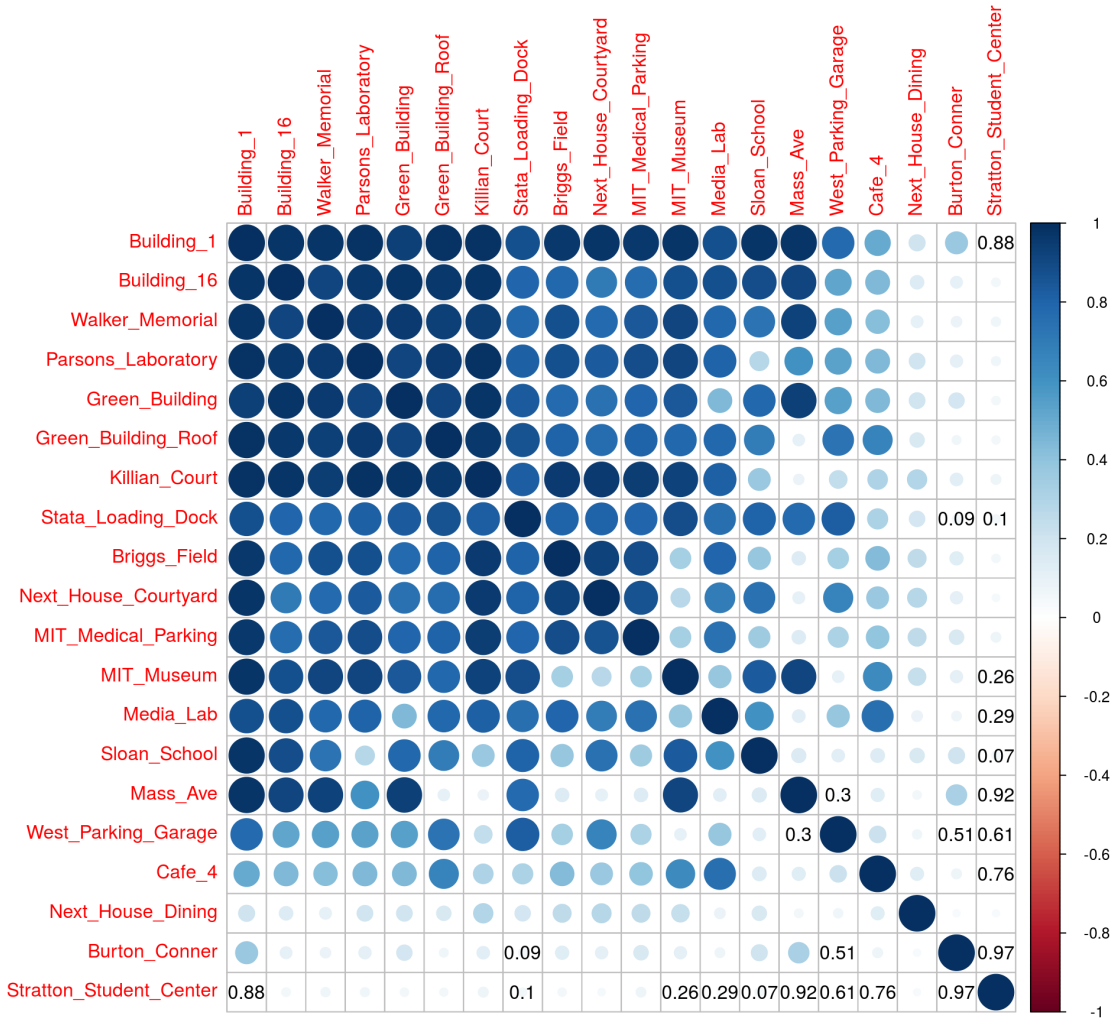


Figure 3.8: Correlations with p-values where not-significant.

Using the correlation value as the inverse of the distance between each node, we can get a type of similarity distance between nodes. Multi-dimensional scaling can use distances between instances to create visualizations that try to respect these distances in a two-dimensional space. The image that results from using Multi-dimensional scaling to visualize the correlation values depicts how clustered different sensors' behaviors tend to be. Our hypothesis was that, if there are indoor and outdoor sensors, it is expected for, at least, two clusters to be clearly visible, and other characteristics should also be visible. The resulting plot can be seen in Figure

3.9.

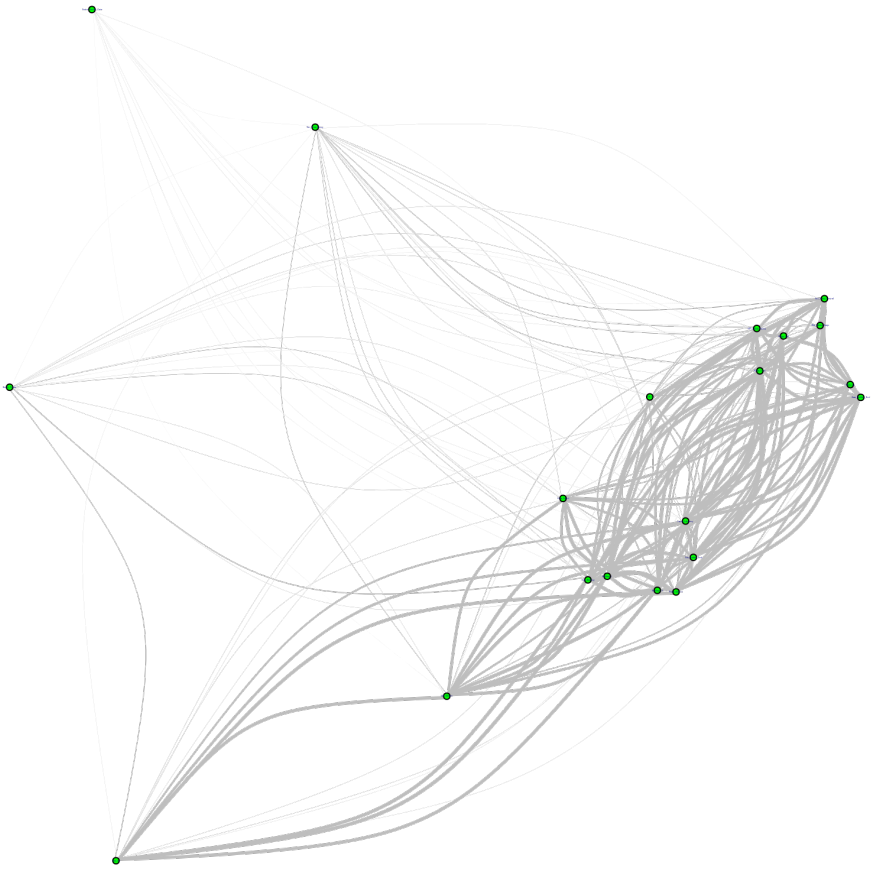


Figure 3.9: Multidimensional scaling.

Auto-correlations

The next step was to analyze the auto-correlations for every node. Because the auto-correlation measures the correlation between an original time series and a delayed version of itself, it allows us to study how cyclic the data is. Figure 3.10 depicts an example of this applied to a *sin* function and Figure 3.11 shows the application of this to the values read by one of the sensors[7].

What we can understand from this is that there is a weekly cyclic behavior, with high correlation every multiple of 7 days. Unfortunately, there is not enough significant data to perform the auto-correlation analysis monthly even though it is very likely to be a strong one. It is worth remembering the auto-correlation value

[7]The blue lines indicate the point of statistical significance - Values between these lines and zero are not statistically significant and those above and bellow are significant.

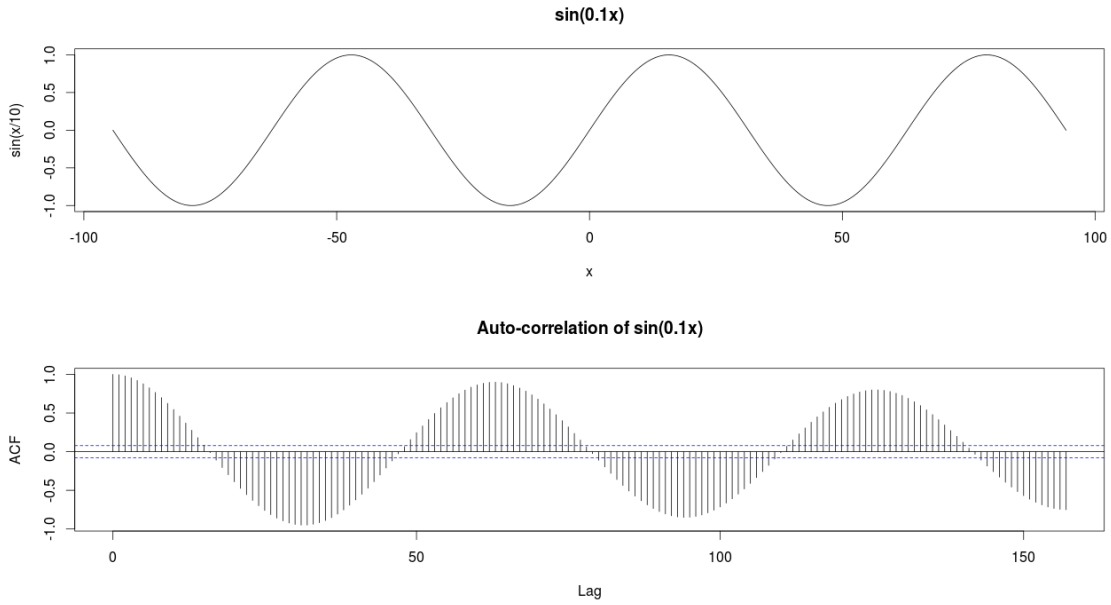


Figure 3.10: Auto-correlation example.

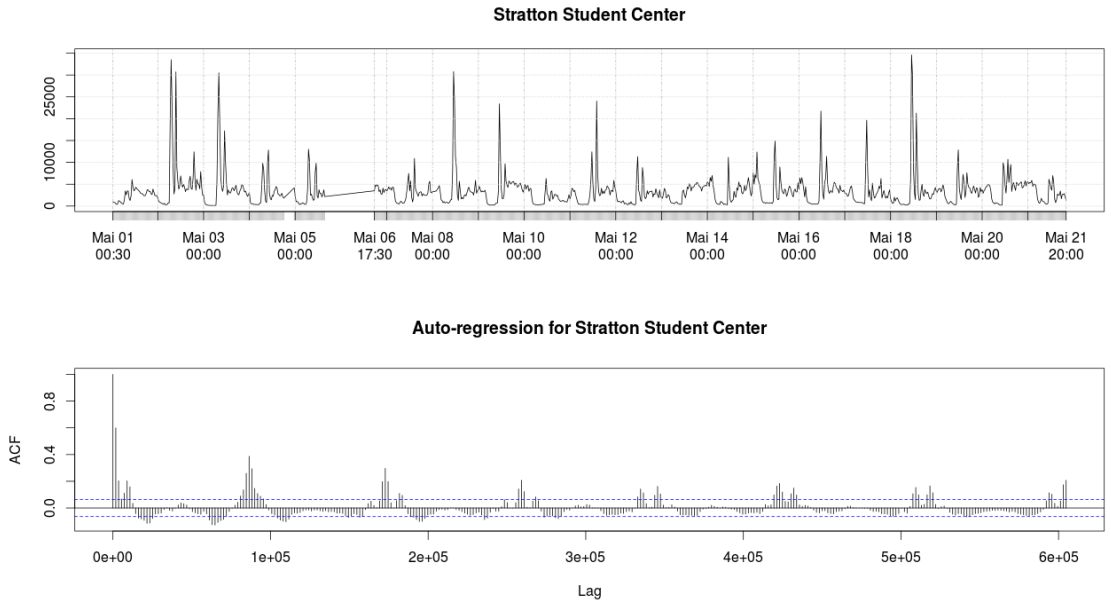


Figure 3.11: Auto-correlation for the sensor located at Stratton Student Center.

will always decrease in time because the data is finite and with each increase in lag, the number of observations decreases.

Cross-correlations

The cross-correlation between different nodes was also analyzed as this could give insights about any lag shared between the nodes. This works in a very similar way to the auto-correlation analysis but, instead of studying the correlation between one time series and its own delayed version, it works by comparing one time series to a delayed version of a second one. Figure 3.12 shows an example of this. As expected, knowing the nodes are highly correlated between each other and themselves have a strong weekly auto-correlation, the cross-correlation graph confirms that the nodes are also highly correlated between each others, even if separated by a matter of some weeks. However, if we plot the vertical lines for each week, there seems to be a decreasing delay between the correlations peaks, which will be discussed later.

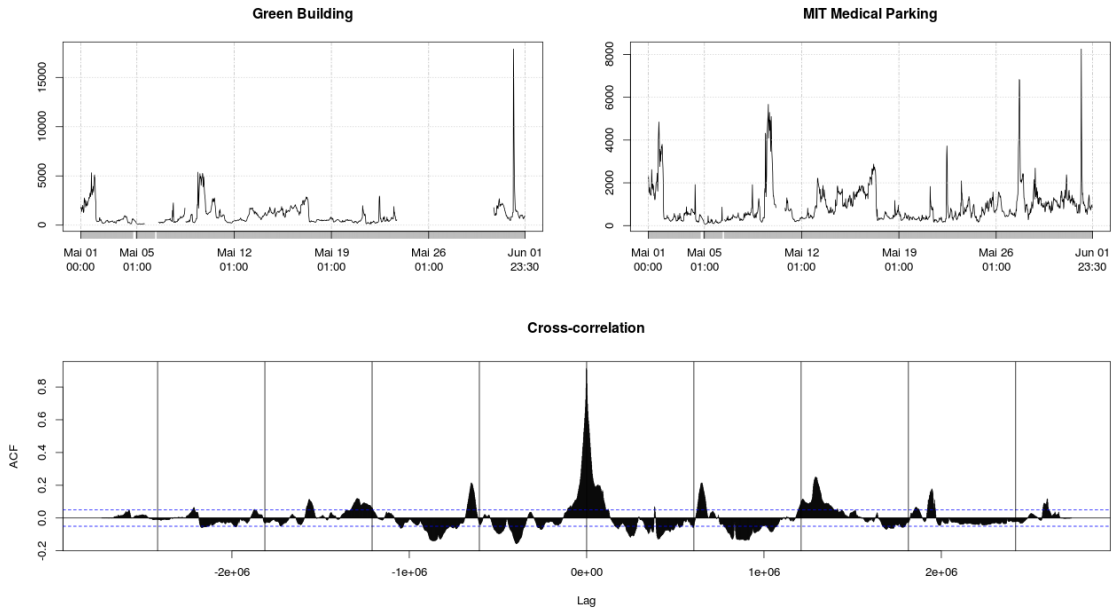


Figure 3.12: Cross-correlation between two nodes.

Eliminated the nodes with insufficient data, the next step was to study their autocorrelations to investigate the existence of cyclic behaviors. This kind of behaviors also indicates the possibility that cyclic features, such as the day of the week, which would contribute to the prediction model. A strong weekly correlation was found, telling us it is very likely for the values to behave in a similar way if the time of the week is the same; intuitively, we can suspect this is due to the pattern drawn each week-end with fewer people on campus.

3.2.4 Descriptive Statistics

Descriptive statistics was also used to summarize each variable distribution, for every node. The first step was to calculate the measures of central tendency: mean, median and mode. The second step was to analyze their five number summary,

a descriptive statistic that provides basic, but fundamental information about the data. It consists of:

- Sample minimum;
- Lower quartile, Q_1;
- Median, Q_2;
- Upper quartile, Q_3;
- Sample maximum.

. This statistic is the precursor of many L-estimators [8]. It is also the core for regular box plots, which are nothing more than a visualization of these values. Histograms were used to better apprehend the distribution of data.

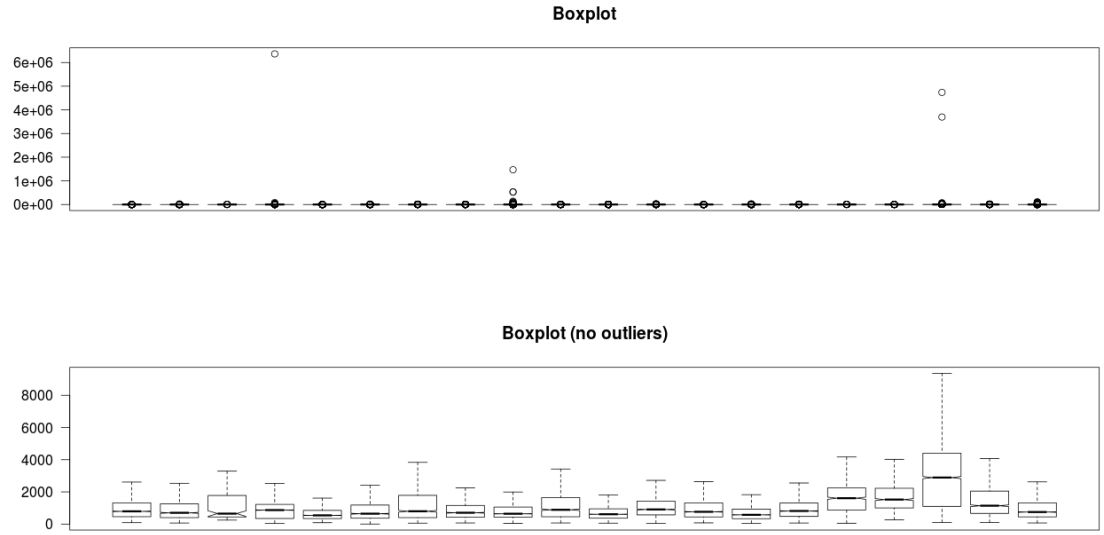


Figure 3.13: Box plots.

From the analysis of Figure 3.13 we can see that the data has many outliers, even though they can be useful in some cases, they can be very hard to model so we choose to remove the top 95 percentile. The box plot for both values, with and without the removal of the outliers can be seen in Figure 3.14.

The final step was to group the values for each half-hour and day of the week and calculate their mean. This way we would know what was the mean behavior for the air quality in any given day of the week or any half-hour for any day. This can be seen in Figure 3.14.

These results were different than the expected. It would be reasonable to think the peaks would appear at very similar times, especially for weekdays, but this was not the case. There seem to be two different patterns spread across the different sensors. One that showed lower values for Wednesdays and Thursdays and other

[8] L-statistic estimators are linear combinations of order statistics, very useful in robust statistics since they are resistant to outliers.

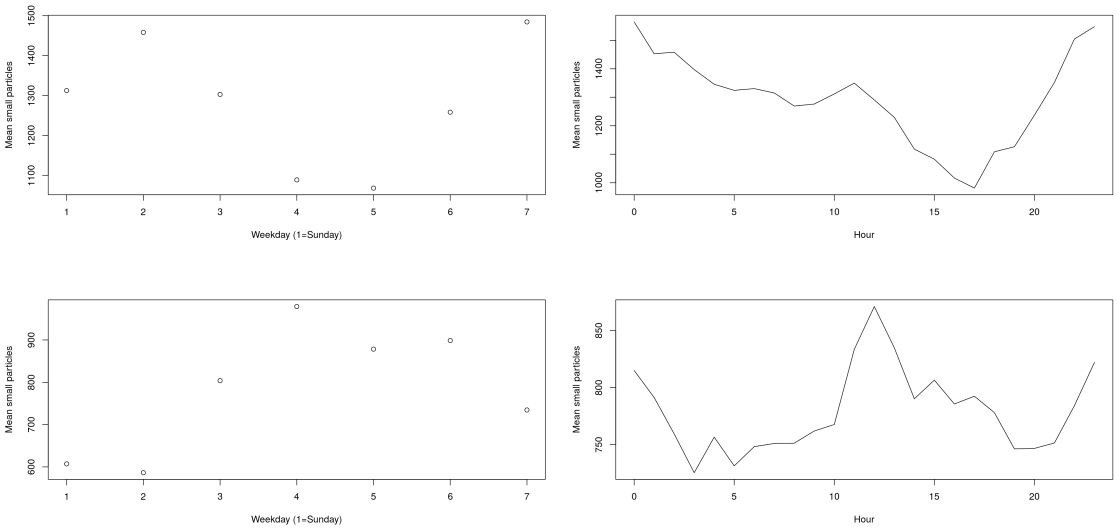


Figure 3.14: Means for two different nodes for different hours and days of the week.

that showed lower values for Saturdays, Sundays, and Mondays. Even though this could correspond to the truth, there was some suspicion about it. After contacting another member of the project, we realized that the sensors used by MIT to develop this network, were made in-house, using readily available parts, one of which was a Raspberry Pi, an affordable, small sized computer that cuts its costs by removing some parts such as the Real-Time Clock, commonly found in computer boards. Because of this, the Raspberry PI needs to use an NTP[9] server to update its clock frequently, based on locale settings, and worse yet, it resets itself if it loses current for a short period of time. We suspect this was a common problem in this dataset as, before, we observed some cross-correlation deviations from the 7 days period, which would not be expected if the clocks were correctly synchronized. We also think that the locale settings are not set up as the same for each sensor, giving a possible justification on why some sensors showed peak values after dinner time and through the night. We expect however that the GP is capable of understanding these problems, provided it has been given enough training data.

3.3 Prediction Model

This section explores the development of the prediction models applied to the Clairity dataset. It starts with the study of different hyperparameters and covariance functions and then jumps into testing the application of active learning using GP's to this dataset. The first application has the purpose to confirm or refute that active learning can provide good results, it does not, however, simulate a realistic application. The second application simulates a real-life application of the model while the third improves on it.

[9]Network Time Protocol.

3.3.1 First Model Application

The first experiment performed was to use Gaussian Processes regression to learn and predict just one node. This would allowed us to choose the best covariance function as well as its hyperparameters. To evaluate the hyperparameters, a cross-validation method called *repeated random sub-sampling validation* was used. This method splits the dataset into training and validation data, in our case 80% and 20% of the data, respectively; then the model is fitted with this training data and evaluated against the validation set. This is done multiple times, in our case, 10 times, and the results are then averaged over the splits. The exploration of the hyperparameter space was then done manually.

The next step was to test the active learning methodology using pool-based uncertainty sampling. To accomplish this, measurements from just one node were selected. The system starts with a sample of 100 instances and then, using the max uncertainty about its prediction, it chooses the next instance to be queried. This application has little value in the real world, however, it serves as a good toy dataset; better than any simple analytic function would, because it deal with real data. In order to evaluate the system's performance, we tested it 50 times against a system that would select any random reading to be learned next. To assure a fairer comparison, the random seeds at the beginning of each run were the same for both the active system and the control setup. The system uses *Mean Squared Error*(MSE) to measure its performance as follows:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2, \tag{3.2}$$

where $\hat{Y}_i$ is a vector of n predictions and Y_i the vector of n true values. Figure 3.15 illustrates the results. Horizontal lines connect the means of the MSE values, while the vertical lines are a representation of the their standard deviation.

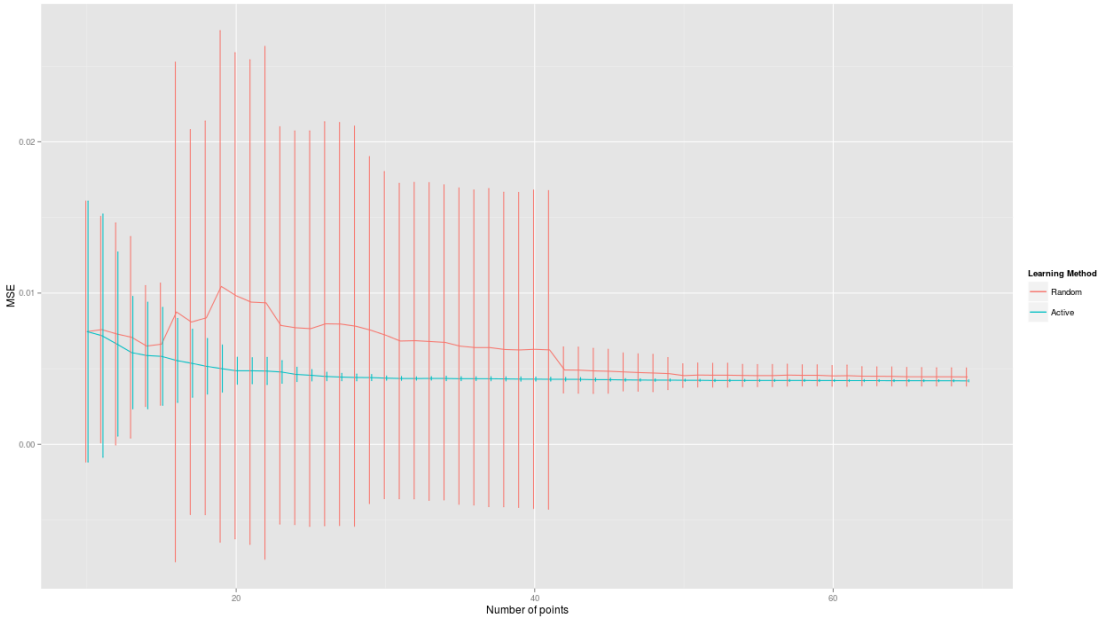


Figure 3.15: Active Learning Toy Dataset.

The output variable used to benchmark the system, small particles, is scaled from 0 to 1, hence such small numbers for the MSE. It is also worth noticing that the standard deviation metric assumes a symmetrical distribution of the values, responsible for the illusion that the random system can be much better at the intermediate state.

3.3.2 Second Model Application

Now that it has been shown that Gaussian Process's uncertainty could be used to develop an active learning system, we could continue ,with some confidence, to simulate the application of this parsimonious sensing method to a real problem.

We developed an experiment that would simulate a real-life example. In this experiment we assume there is the need to evaluate the air quality around the MIT campus, however, we added some restrictions and assumptions:

Limited number of sensors: Lets assume the management only authorized a single sensor for long term use, either because it was too expensive financially or there was a very high maintenance costs associated to keeping various sensors;

Availability of each sensor: We also assume that we know when a sensor is going to be able to read more than 50% of the time;

Learning phase: There is a learning phase, where various sensors would be distributed around the campus and collect information for every sensor. This would only happen on the first day;

Portable sensors : The sensor can be moved between 7 different places.

With these restrictions, the problem was for the following days, to have a strategy to choose which sensor location to use.

A naive approach, would be to randomly select the sensor's location for the next day, while another one could be to number the sensor's available placements and cycle through them, without repeating one until everyone was used the same number of times. Even though these seem very viable solutions, our objective was to test if we can use active learning to select which one would be the best place to move the sensor to by selecting the place we are less certain about.

We finally tested three different selection methods, on all of them, only places where the sensor was available more than 50% of the time could be selected:

- The first method, is the random selection of the next place to be explored. This method does not guarantee that the system will not choose repeatedly the sensor placement every day, however, the odds of this happening, even assuming each place is available for every day (with more than 50% of the data) are extremely low;

- The second method, assumes that we always know, not only if that place is available for use the next day, but also how much of its data it is available, i.e. the percentage of the data available for that day. Using this information the system would choose to maximize the known data, selecting the placement that would retrieve more information for the following day;

- The last method, uses active learning to select the next sensor placement. To do this, the system, using the known data to learn the GP's, would try to predict the sensor values and their confidence for the next day. Then, these predictions would be grouped by sensor placement and the mean of the confidence for each group is calculated. Finally, the system selects the placement where the confidence is lower and is available to be explored on the following day.

To evaluate this model, the system would predict the values for the following day and compare them against the observed values. Finally these results would be plotted, as can be seen in the last section of this chapter.

The Matlab library that we used, GPML, has a function called *minimize* which serves to train the hyper-parameters, minimizing the negative log marginal likelihood. We have found this search to be extremely local for our application and we wanted to find a way of automatically perform a broader search over the search space. While reading about different algorithms for Hyper-Parameter Optimization, we found about Trees of Parzen Estimators and, soon, a library, named hyper-opt, that implemented them, in Python. To use this library we simply needed to define the search space and the fitness function. Our search space was defined by the different likelihoods, inferences, covariance functions and their arguments, while our fitness function was the performance of the active learning system, that measured, using the RMSE, would represent the quality of the prediction for the last day. Having defined all of these, we would then try to find the best hyper-parameters for a short period, 4 days of the dataset. These would serve as a global search for hyperparameters that would then be used at the system initialization. The function *minimize* would then be used each time new information was added to the system, in our case each time the system gathered information for one more day. A sample code for using the hyper-opt library can be seen in Code 3.1.

Listing 3.1: Definition of Hyper-parameters for Hyper-opt

```
space = hp.choice('model_likelyhood', [
    {
        'likelyhood': 'likGauss',
        'inference': hp.choice('g_inference', ['infExact', 'infLaplace']),
        'ell': hp.uniform('g_ell', 0.01, 2),
        'covFunc': hp.choice('g_convFunc', [
            {
                'covFunc': 'covMaterniso',
                'covArg': hp.choice('g_d', [1, 3, 5])
            },
            {
                'covFunc': 'covSEiso',
            }
        ]),
        'sf': hp.uniform('g_sf', 0.01, 2),
        'sn': hp.uniform('g_sn', 0.01, 2)
    }])
best = fmin(test_args,
            space=space,
            algo=tpe.suggest,
            max_evals=300,
            trials=trials)
```

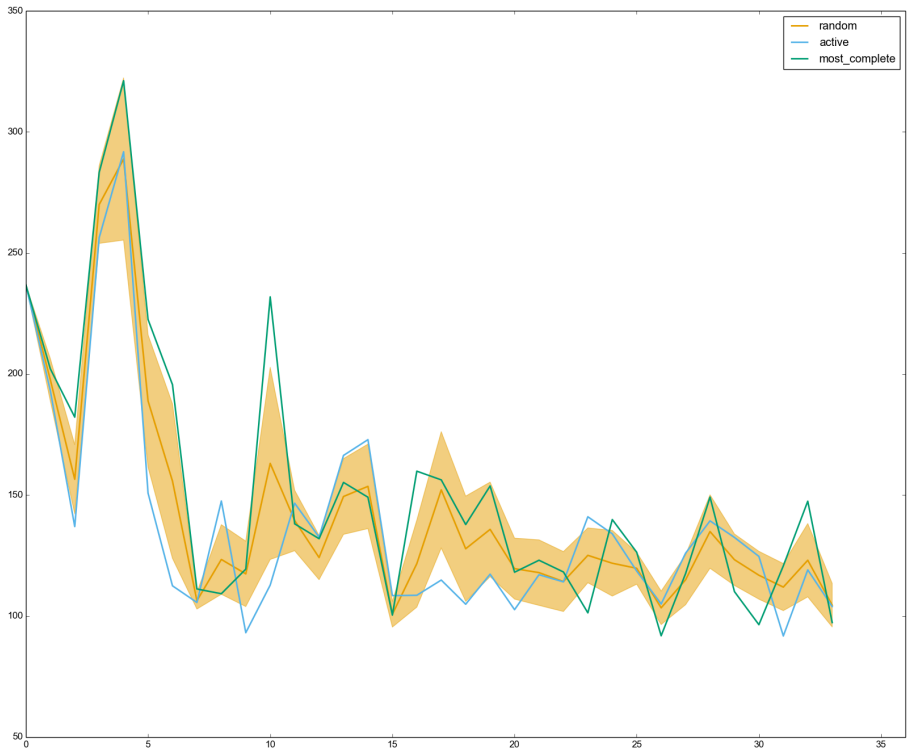


Figure 3.16: Evolution of RMSE when evaluated against the next day prediction.

As we can see from the results shown in Figure 3.16, active learning is superior to both random learning and learning the most complete node, but not every time and not by very much. The inconsistency of the results for each day is also very visible with high local extremes on the first days and some other extreme variations on the tenth and fifteenth days. At this point, we think that active learning has more potential then it seems to have in this example, and using the following day to evaluate the algorithm, may not be the best option since there are big variations of behavior and sensor availability from one day to the next.

We also observe that consistently selecting the sensor from which we have more data can actually be detrimental to the system. If there is a sensor with no missing data, it will always be selected, and even though the amount of training data available to the system is greater, its quality is not. Furthermore, the system is evaluated against all sensor locations, so if there is a sensor that is very incomplete, the active learning system will try to learn from it (if available more than 50% of the time for that specific day) since it is less certain about it. On the other hand, selecting the most complete sensor will always ignore this one, leading to worse end results.

3.3.3 Third Model Application

The third model application was very similar to the previous one, and, if one analyzes its results, its motivation can be easily understood. The first reason was that not every day had sensor information for every location, so sometimes we would be just

benchmarking the system against a limited dataset. Another reason was that some days were just more unusual than others, which would be difficult for the model to predict, resulting in very high error values.

To solve this evaluation problem, 5 days, in the future, where every location was represented with at least more than 50% of the available data, were selected, and the system was ran against it. The system still used its prediction of the following day to select which node to explore, the only difference is that now, it is evaluated against 5 unknown days.

For this experiment, we used only the top five most complete sensor locations for one month and five days. Again, the first day was used as learning phase, where sensor information would be gathered from all the available sensors. This was done 29 times, with the system trying to predict those 5 days each passing day.

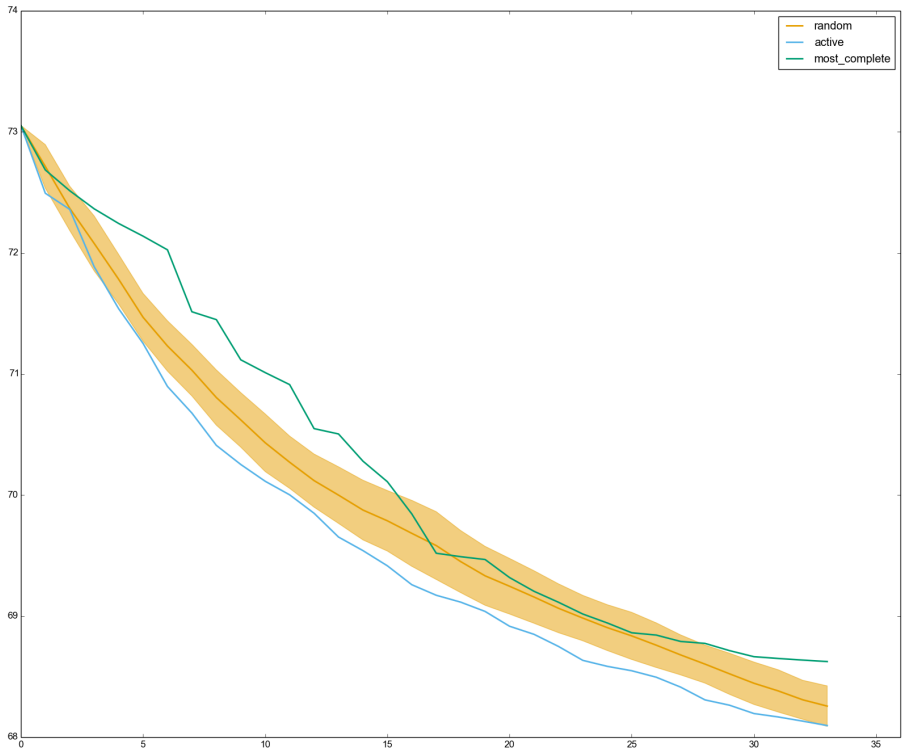


Figure 3.17: Evolution of RMSE when evaluated against the fixed set of days.

The improvement on the evolution for the Root Mean Squared Error is visible in Figure 3.17. After changing the evaluation function, not to minimize the RMSE for the last of four initial days, but to minimize the RMSE for 5 different days in the future, the evaluation became more stable resulting in an immediate and drastic impact on the system performance, from the first day, decreasing from about 240 to 73. The evaluation curve is also a lot smoother now. The superiority of the active learning system is visible throughout the graphic, after the fourth day, and we can affirm with 95% confidence to be better against a random selection of nodes. The effect of selecting the most complete nodes is also visible from the third day

onwards, where the RMSE begins by getting worse than the random learning mean from which the system can not recover until the sixteenth day.

3.3.4 Discussion and Improvements

The most notable conclusions are that Active Learning proved to be superior to other methods in an environmental sensing application; and that, as we initially expected, choosing the most complete node does not result in better results than choosing a node randomly. We also think that these two last models could be improved even further by exploring different ways of evaluating the system's performance, such as leave one out. In that case we would run each setup the same number of times as the number of days, and, each time, remove one day from the training data and use it as the evaluation, calculating the mean results after all the tests. These tests were not ran because Gaussian Processes demand very high computational power and each would take about 30 times more processing time.

We also thought of adding a very simple memory to each system, this memory would prevent the system to explore the same sensor placement twice in a row. The advantages are obvious for the random selection, as it would make impossible for the system to choose, for example, the same sensor everyday, which, although unlikely, is possible. As for choosing the most complete sensor, it would also prevent problems when a predictable node has consistently a high percentage of available data. The effect could be even more drastic for the active learning method. The problem of applying the GP's blindingly in this problem, is that they can choose to learn from the worst nodes, the ones that give very noisy readings, have low routine behaviors, or simply little information for all days; in this case, selecting the most difficult sensor placement to predict would actually result in worse performance overall.

Chapter 4

Event Popularity Prediction

The goal of this experiment was to test the application of the Parsimonious Sensing Methodology to predict the popularity of an event, based on the number of hits returned by a search engine. By doing this, our objective was to predict with some success how popular an event would be based on its characteristics, reducing the number of searches that was needed to do to a minimum.

Before jumping to the main question, one should focus on other, simpler, questions like:

- How hard/easy is it to predict search engine hits?
- Are topics enough for this task?
- Does the number of topics have an effect on the quality of the predictions?
- Would it be worth to create a model for each query size? I.e. one model for queries with two words, three words and so on?

The first question has already been answered before in this document: predicting the number of hits can be very complex. We expected this experiment to be a challenge as many, if not all, of the papers referenced in Section 2.3, have shown how hard this problem can be to tackle. From restrictions to the possible queries that can be done to very unpredictable behavior, varying from search engine to search engine; many were the problems faced in this chapter. However, we still wanted to try an experiment with the objective of predicting the number of hits for a specific query, our reasoning was that, instead of using single words to analyze the queries, we would try to limit the dimensionality of the problem by using topic modeling. Furthermore, we would limit the application to a very specific domain: events such as music concerts, expositions, workshops, etc.

The last three questions are answered throughout the development of the experiments described in this chapter.

4.1 Dataset

For this experience, a dataset was designed from scratch. We started with a list of 4,829 events that happened in Singapore. We then defined the format of a query as the event name plus "Singapore". This would result in the same number of queries to be run, which would in turn result in the number of hits on Bing search engine.

Query	Number of Hits	Processed Query	Number of Words
We Are Like This Only!	214,000,000	[like]	1
Digital Photography I	8,390,000	[digit, photographi]	2
The Little Red Hen Singapore	33,000,000	[littl, red, hen]	3
Singapore Toy, Game & Comic Convention	3,280,000	[toy, game, comic, convent]	4
Dividends Don't Lie Singapore	30,900	[dividend, lie]	2

Table 4.1: Dataset with number of hits and number of words for each query.

To these two rows, query and number of hits, we added another for the number of words of each query. To calculate the number of words we removed stop words[1], as this is common practice in information retrieval systems [32, 33]. The remaining words were then stemmed using a lemmatisation algorithm and added to the dataset in a new column.

Table 4.1 shows the head[2] of the file with one added column for the words that were not processed.

4.2 Data Preprocessing

The next step was to apply topic modeling algorithms to the queries, more specifically to the filtered and stemmed words. We experimented with both LSI[3] and LDA[4] with the number of topics ranging from 2 to 99.

By defining the number of topics when we apply this algorithm, we get the topic distribution for each one. In other words, for example, if we have 3 topics, for each query, both of these algorithms will return three numbers. Summed up, these are equal to one and each of these numbers will correspond to how much a query belongs to a specific topic.

The main purpose of this was to extract semantic meaning from the queries and try to understand if words associated to the same topics, and so, associated between each other, could have similar search results. For example, if the word *live* was strongly associated with the word *music* our hypothesis was that they will have strong correlation in terms of search results, and this would be enough to make some kind of predictions. This would work, in a sense, as feature reduction, and we expected this to help the prediction model.

Our final, preprocessed dataset was then composed of 4,829 queries and their corresponding topic distributions, a sample, for three topics can be seen in Table 4.2.

Query	# of Hits	Bag of Words	Topic Distributions	Stemmed Words	# of Words
We Are Like This Only!	214,000,000	(0, 1)	[0.167, 0.189, 0.643]	[like]	1
Digital Photography I	8,390,000	(1, 1), (2, 1)	[0.125, 0.755, 0.120]	[digit, photographi]	2
The Little Red Hen	33,000,000	(3, 1), (4, 1), (5, 1)	[0.825, 0.087, 0.088]	[littl, red, hen]	3
Singapore Toy, Game & Comic Convention	3,280,000	(10, 1), (11, 1), (12, 1), (13, 1)	[0.068, 0.862, 0.069]	[toy, game, comic, convent]	4
Dividends Don't Lie	30,900	(16, 1), (17, 1)	[0.724, 0.118, 0.157]	[dividend, lie]	2

Table 4.2: Dataset with topic distribution and bag of words added, for each query.

[1]We used Python's Natural Language Toolkit 127 stopwords list. If this proved to be insufficient, we were ready to use the Information Retrieval Group's 319 stopword list.

[2]The first 5 rows.

[3]Latent Semantic Indexing.

[4]Latent Dirichlet Allocation

4.3 Exploratory data analysis

We started the data exploration by studying the number of results for each query length. We defined the query length as being the number of words, not the number of characters, which can be seen in the previously presented tables, including 4.2.

To do this, we group the queries by length size and plotted both mean and median (Figure 4.1 this is an example where the difference between the two shows with very different results, especially for the query length of one.

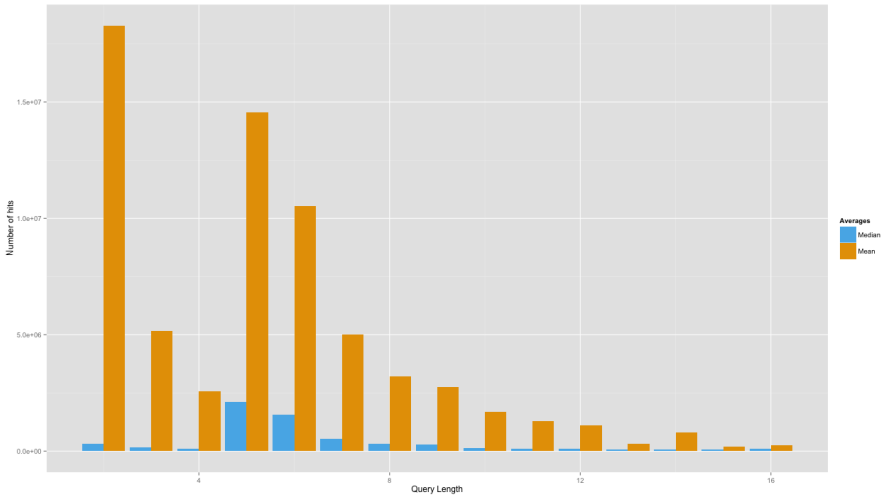


Figure 4.1: Comparison between the two averages.

This result lead us to believe there was a peculiar distribution on the number of hits, and, after removing values above the 90% percentile, we plotted a boxplot to help us understand better the results, which can be seen in Figure 4.2.

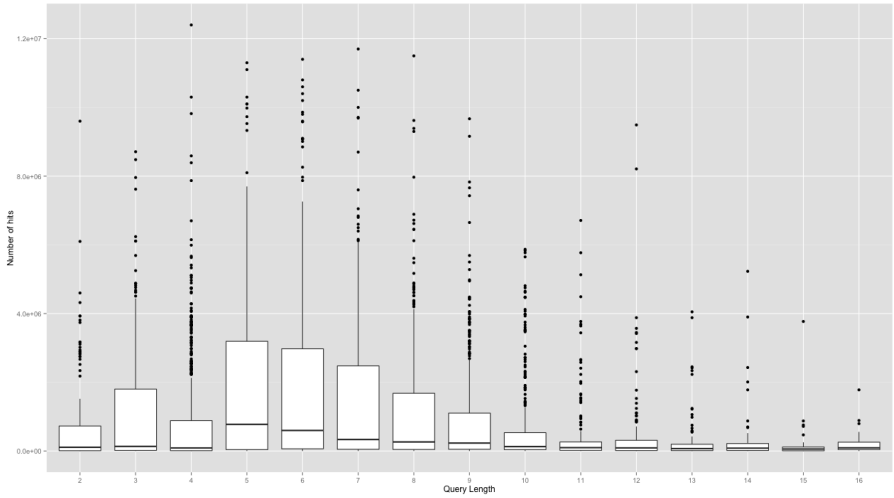


Figure 4.2: Boxplot for the number of hits, grouped by query length.

From these results we can conclude that the distribution is not normal, in fact, we can observe from Figure 4.3 that the distribution of hits for the number of results for 8 word queries is one long-tailed distribution.

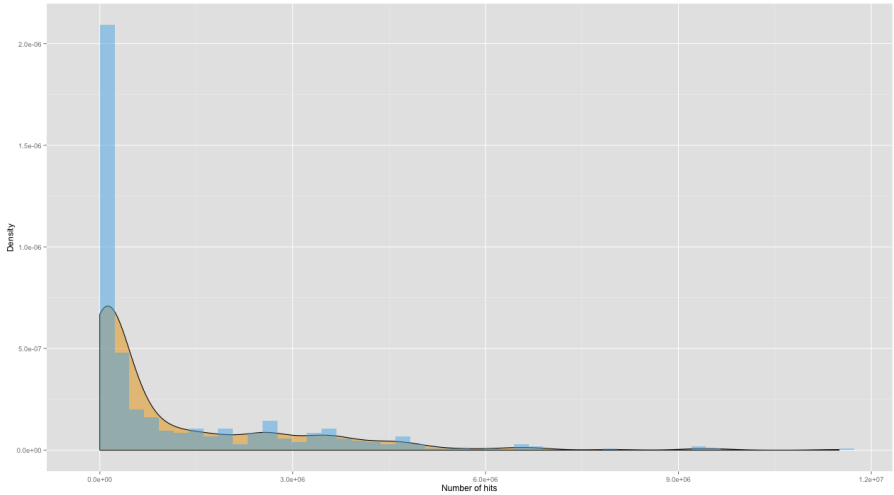


Figure 4.3: Histogram and approximation of the probability density, distribution for the number of results, 8 word queries.

Regression algorithms, in particular Gaussian Processes, can suffer from this kind of distributions without specific data treatment such as output transformation, e.g. the application of a function function such as log and its inverse to the output of the system; or by using a specific kernel that takes this distribution into account.

We also wanted to see what was the effect of applying LSI and LDA for queries of different sizes. We found this results to be very peculiar, and it would be interesting to study this phenomenon in greater detail.

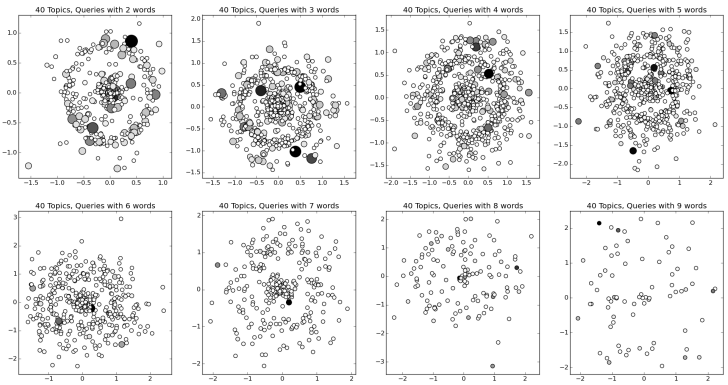


Figure 4.4: Distribution for the number of results, for 8 words queries.

A pattern starts to emerge when LSI is applied to the queries. This pattern becomes even more evident when LDA is applied to the data as can be seen in Figures 4.5 and 4.4. This pattern shows a tendency for the creation of "similarity rings" which are the same number as the number of words of the queries. This may direct us to believe that when only two words can vary, a small variation on either one would have a visible impact on the query proximity with other queries. The number of topics also had a visible impact on the proximity between queries. It would seem like the feature space would be divided between the number of topics, so if we used 8 topics, queries would most likely align in lines that would have one of

eight angles in relationship to the center of the image. When using a large number of topics such as 40, these lines are no longer perceptible and rings begin to manifest more and more. We also found that using LDA instead of LSI resulted in more separation for each document from topic to topic, which can be seen in Figure 4.5.

A visualization for only four topic can be seen on Appendix B and shows that, in fact, these shapes are not rings but not very different from deformed polygons with the same number of vertices as the number of topics.

We suspect this to be an artifact caused by the visualization algorithm, Multidimensional scaling, which as been used before, in Section 3.2 to aid with the visualization of correlations between different sensor nodes.

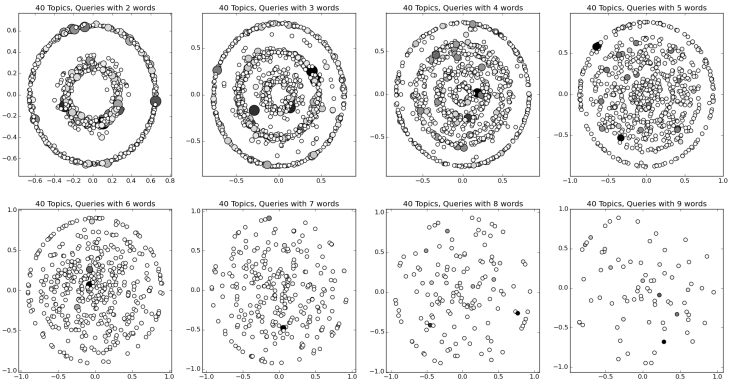


Figure 4.5: Distribution for the number of results, for 8 words queries.

4.4 Prediction Model

We followed this exploratory analysis with the application of Gaussian Processes to predict the number of hits for a certain query.

Since we didn't know *a priori* what would be the best number of topics, we started with a simple analysis where the dataset was split in a train and test group, with 80% and 20% of the data, respectively.

We tested using 2 to 99 topics, the results can be visualized on the Figure 4.6.

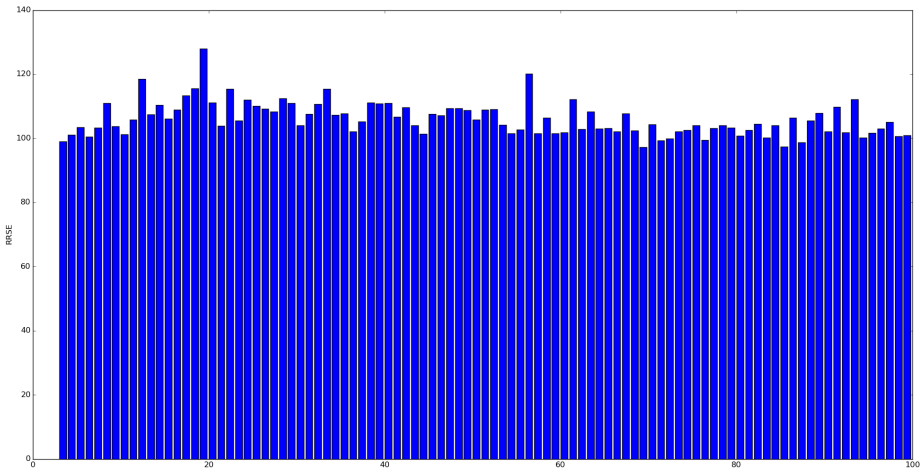


Figure 4.6: Performance for different number of topics.

4.5 Results and Discussion

These results confirmed how hard it can be to accomplish satisfactory results when predicting search engine results, using topic modeling for the specific task of predicting search engine hits for events happening in Singapore. Because of this, we came to the conclusion that if we wanted to analyze events using context from the web, this was not the right direction and we should explore other possibilities. It could be interesting to try this same experiment but to perform the topic modeling over a big corpus and then see the topic distribution for the queries. The fact is that only very specific keywords are usually used to name an event and that can make it hard to establish connections between new, less, common words. Another improvement would be to test this with a larger dataset, as it is possible that this problem is so complex that little less than 5,000 events is simply insufficient to perform this type of predictions.

Chapter 5

Event Detective

Building on the experience from the previous chapter, we revised our objective to focus on setting up a system capable of identifying large movements of people to a certain area and the causing event.

Having a system that could solely identify unusual behavior on the streets would be particularly interesting to a traffic administrator or law enforcement agent. This however would still force the user to try to identify the causing event on a search engine or other alternative. Keyword based search engines often return too many results, leading to information overload [54]. Also it would be very time consuming for a traffic administrator to go through all zones and identify where there were anomalies. If this system could also provide, quickly and automatically, the context in which it happened, it would be a great advantage.

In this chapter we present *Event Detective*, a dashboard system that would inform its user of the events happening in that place at that time, resulting in higher quality adaptations to the city's transportation system, while avoiding the unnecessary search for events, effectively resulting in a a type of parsimonious context sensing. The question would then be: "Can we develop a system capable of identifying anomalies and the events responsible for them?".

5.1 Anomaly Definition

Since we wanted to study different traffic anomalies, we first had to define what an anomaly was. A very obvious one would be to identify any kind of traffic jam, i.e. where cars would be moving at a speed unusually slow for that area. However, this could be due to an unpredictable event such as an accident or a fire. Instead we defined an anomaly as a place where there would be an exceptional number of drop-offs.

5.2 Dataset

The first step to answer to this question was to get traffic data, or anything that would represent it in a satisfactory manner, and then how to get context automatically and identify the culprit event.

Singapore is renowned for its unusual number of taxicabs, having, as of June 2015 a total fleet of 28,686 taxis, operated by six different taxi companies and almost 200

independent taxis. These taxis usually have two assigned drivers, each with a 10 to 12 hour shift. There are two major reasons for taxi demand to be so high in Singapore. The first is that according to Singapore's law, to own and use a car in Singapore, one must buy a Certificate of Entitlement (COE), these are bought through an open bid uniform price auction which can drive the price of a basic Toyota Corolla to prices in excess of 100,000 SGD. In fact, just the open category COE itself has peaked in January 2013 costing 97,889 SGD at auction. This would still not have the same impact if these COE were not only valid for a period of 10 years, at which point they must be extended.

Luckily for us, a major taxi company in Singapore provided us with some of their private data, which, unfortunately, cannot be shared outside the SMART's workforce as it is protected under a non-disclosure agreement. The information shared here protects the privacy of the company, whose name shall not be disclosed.

This 154.4 GB dataset is composed of 76 different log files, which describe the positions and status for all the vehicles of a major Taxi company in Singapore. Each file represents a unique day, from 2012-06-01 to 2012-08-15. Each of these files has about 35 million rows, each of which composed by: Time stamp, plate number, latitude, longitude, speed in kph and taxi state.

The field that defines the taxi state can describe different states on the taxi meter, these are:

- FREE - Vehicle is free to take passenger;
- POB - Passenger on Board;
- OFFLINE - Driver has logged off;
- BUSY - Vehicle is not free to take passenger;
- STC - Soon to Clear, Vehicle will become free in short time;
- PAYMENT - Cashless payment in progress;
- ONCALL - Vehicle is on hire. Not available for street hire;
- BREAK - Vehicle is not free to take passenger;
- POWEROFF - Not used;
- ARRIVED - Vehicle has reached the pickup location.

5.3 Preprocessing

It would be extremely hard to use the dataset as it was so we had to find a way to extract representative information from it, which is described next. It was very important to reduce the amount of information and try to reduce the dataset to a manageable size. We start by extracting the most important information, which was for us the taxi trips and then we clean any invalid data that we could find and perform statistical analysis over the remaining data.

5.3.1 Trip extraction

The dataset is composed by raw spatio-temporal data for each taxi. If we assume a trip the time and space from when a specific client enters a taxi until the time he leaves, we can split this data into separated trips.

This script would simplify further analysis by aggregating the data relative for each trip in eight different fields:

- origin_latitude;
- origin_longitude;
- start_time;
- destination_latitude;
- destination_longitude;
- end_time;
- duration;
- taxi_plate.

To do this, the script would go through each row, sorted by reading time and keep track of when and where, for each plate number, the status changes from POB [1] to any different one.

I found some dataset errors while performing this analysis, for example, some trips had an exceptionally big or negative duration. These are likely to be the result of a system left running or an updated internal clock respectively. Fortunately, these account for only 285,022 of 64,231,651 trips, which leaves us with 99.56% of the dataset.

Another script removes this invalid trips by filtering the ones with a negative or above one day duration.

5.3.2 Statistical Analysis

We would consider an event, when at some place and specific time of the day, there was an unusually high traffic of taxi drop-offs. As both the geographic and the time features for the trips are high in precision, if we wanted to perform statistical analysiso n them, we would have to quantized them into lower frequencies. Similarly to what we did in Chapter 3, values were averaged for each half-hour of each day.

As for locations, this problem is not so simple to solve. Except for specific places and exceptional times, taxis are limited to the streets, and, by doing the same type of analysis, i.e. dividing the space uniformly, we are ignoring many of the characteristics underlying in this kind of systems: city organization, event centers, terrains and others.

There has been some work using cellphone tower-based locations to identify places with greater population density for the identification of events. In this case the city was divided into 319 cells using a Voronoi tessellation [19] based on the cell tower's localization [11]. Ideally, one could use something similar to this, taking into account the specific case of transportation. To do this, we used a dataset from LTA [2], that would help us solve this problem. This dataset divides the country of Singapore in 1,092 areas, and it was developed having all the characteristics of Singapore streets and traffic into account. This zones are defined as a set of polygons with their respective names and other information. We could then merge the information from the two datasets and identify to which zone a specified coordinate pair would belong to. These zones can can be seen in Figure 5.1, with each color representing one of the 7 different regions.

With the definition for these zones, the same way a histogram aggregates different values in some bins, we can now aggregate values by their zone, day and half-hour.

[1] Person on Board.

[2] Land Transport Authority.

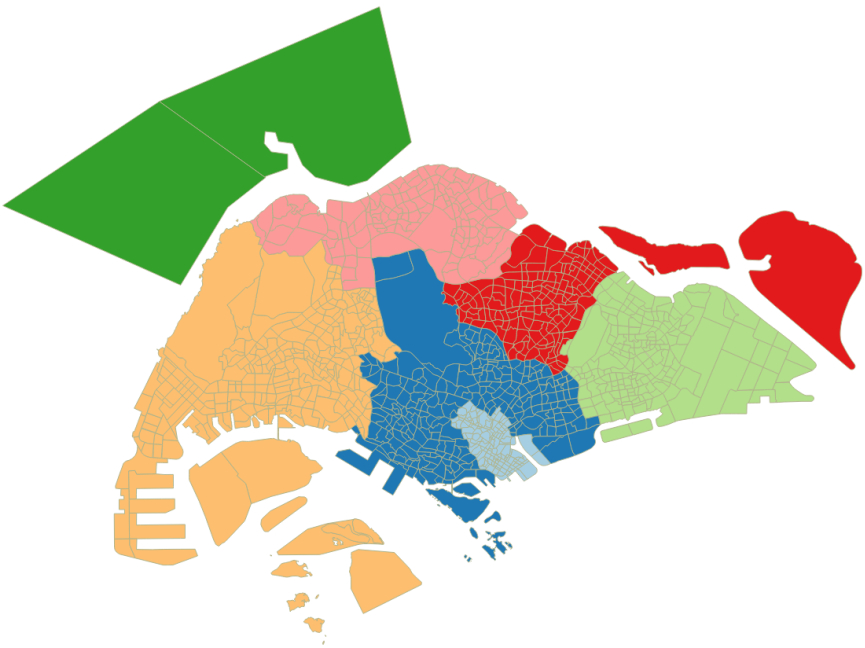


Figure 5.1: Singapore Zones.

This, in conjunction with some statistical analysis, would make us able to identify what we first defined as an anomaly: a particular area and time where there would be an unusual number of drop-offs. The next step was to define, what was, specifically, out of the ordinary.

The first rule we used to define an outlier was very simple, anything above 95% of the other values would be considered one. There is although a problem with using this rule: since we are using the 95th percentile, we consider always 5% of the data as an outlier, even when it has very little semantic value. To limit the number of results, we filtered only days where the outlier would be at least twice of the 95th percentile and above 200. First we aggregate the data by day of week, zone and half-hour. Since we only have 77 days this means we only have, 11 readings of each type, e.g. 11 readings for all Mondays, at 10h00m in a specific zone. We then went through all these combinations, and to see if one is an outlier, we exclude it from the analysis, calculate the percentiles for the others in the same day of the week, zone and half-hour and compare. Initially this analysis would take about two weeks to run, however, I was able to reduce that to just little more than a day, more details can be found on the Appendix D.

5.4 Exploratory data analysis

Now that we have some statistics we can visualize both the routine behavior for each zone on a specific day and the specific times where the drop-off values were greater than expected, Figure 5.2 is example of this.

This image tells us that, usually, this place does not have more than about 40 drop-offs throughout that day of the week, however, at this specific date, at about 15h30 (31th half-hour) there is a rise in drop-offs, going well above 95% of what was seen on any of the other days, exceeding the 350 mark for drop-offs in some of the

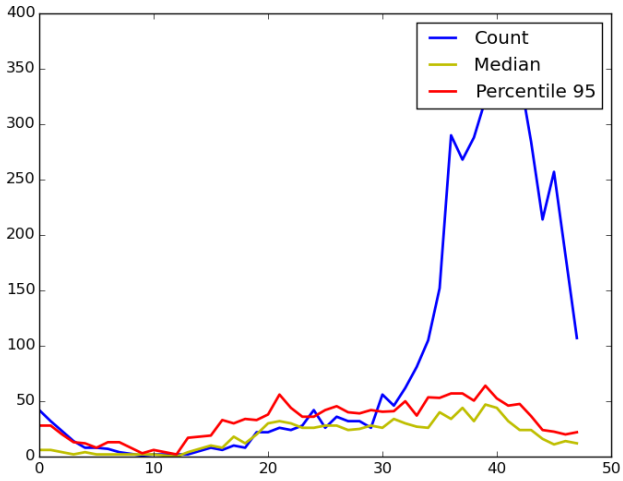


Figure 5.2: Routine behavior for a specific zone.

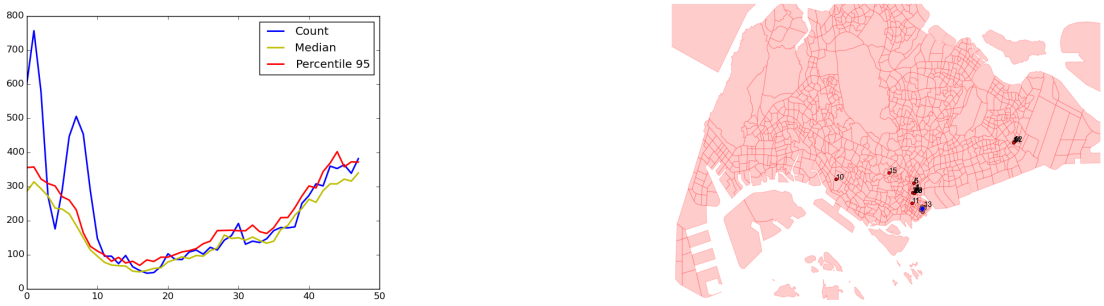


Figure 5.3: First dashboard.

30 minutes split. We can state with some confidence that this is the effect of an event happening in the area, that peaked around 8 pm.

5.5 Manual Event Identification

The following step was to identify a set of anomalies and the correspondent events that caused them. By identifying to events manually, we could get a greater understanding of how the human mind works when trying to solve a problem like this using various sources, and we get the results for comparison with an automated system.

To help with this task, I developed a dashboard application that shows the statistics of a specific area and a visualization of the area, a first version of this can be seen in Figure 5.3.

This system turned out to be insufficient to identify most of the events found, as the zones would be hard to identify. To try to solve this problem, the dashboard was updated with a new map, now using OpenStreetMaps for the visualization of the drop-offs. This would have a major impact on the dashboard usability as the user now could not only identify the area in question but could also zoom in and out and move in all directions, which would simplify the identification of each individual

drop-off. Figure 5.4 shows an example of a what the user could see when analyzing a specific day near the Singapore Flyer. Later this system would also show the drop-offs that happened during the outlier in a different color than the customary drop-offs. The dashboard also showed the place where the passengers got inside the taxi, as this could help identify events where people parted from one place to another, e.g. private companies dinner parties.

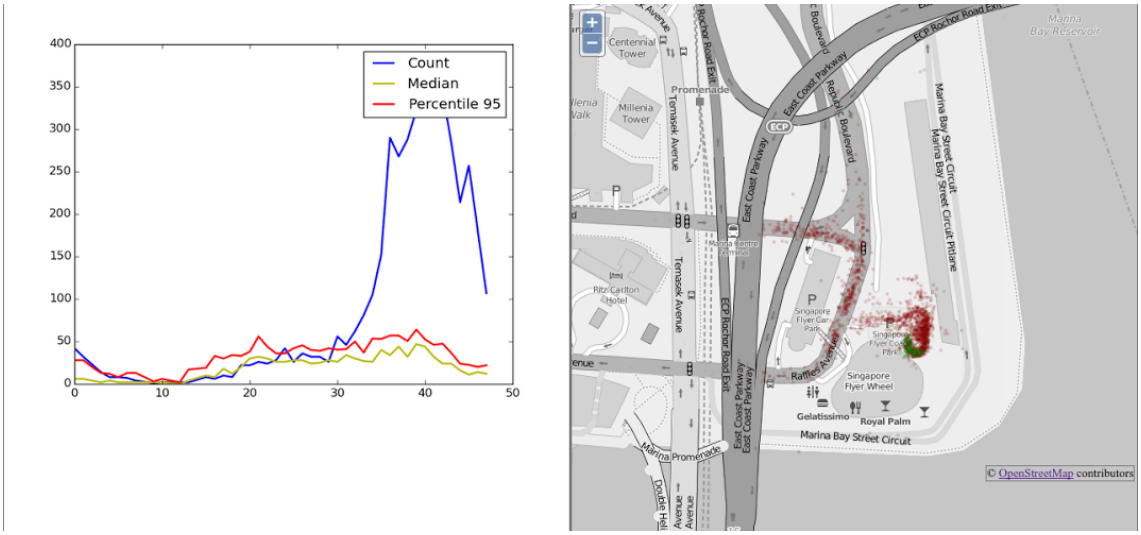


Figure 5.4: Improved version of the dashboard.

5.6 Manual Identification

By looking up the events manually we understood that a person can employ different techniques to identify the anomaly causing events.

An option would be to get a list of events that occurred in the period showed. Then, we could look up each event one by one and see if any one would be close to the particular area with unusual drop-offs. This however had limited success as most of the event aggregation websites cannot capture all events happening in Singapore, and even the listed events have, frequently, incomplete fields such as starting time or venue.

Another, more general approach would be to look up manually for what we thought was a POI[3] and for that specific day on Google or other search engine. This would generally include the results from the previous option as well as new sources. However, some false positives are also more likely.

To evaluate our automatic model, we did this searches manually and created a dataset with about 30 events identified in a supervised way. Once an event was found, we would save the pages that identify, creating a dataset with what we thought, was the name of the Point of Interest, the date, time, name of the event and websites that identified it correctly. Part of this dataset would also be used in the development of the first automatic system, by using the manually identified POI's.

[3]Point of Interest.

Figure 5.5: Outliers showing the Singapore Expo and the Marina Bay Sand Hotel.

This dataset would then serve as ground truth for the evaluation of the new, automatic systems.

5.7 Automatic system / Model deployment

Using what we learned from the previous process, we then began implementing an automatic system that would create Google queries, using text describing the place and time of an anomaly.

5.7.1 POI's identification

To implement this, the first thing we needed was to convert a zone to something searchable on Google. Manually we could identify on the map places that we know a priori that were POI's however, a machine has to do this automatically.

For this, we first needed a way to select representative points for the drop-offs as a whole area should be too wide to identify the event venue. After some thoughts and looking at some maps, like the ones seen in Figure 5.5, clustering algorithms came immediately to our minds when looking at some maps, they seemed the obvious answer.

Different clustering algorithms were tested: density based scan, k-means, affinity propagation and a custom algorithm. By selecting one, we would have a way to extract interesting points.

Our first test was to apply the density based clustering algorithm. It seemed possible to apply it correctly, since we wanted to look for a place where the density of drop-offs was higher. However, it also carried some problems:

- The density based scan does not have centroids, so we would need to develop a way to identify which point from a cluster would represent it. Simple solutions, such as using an arithmetic mean, are not successful as there is no guarantee the points will be distributed in a circular way, leading to incorrect placement of the centroid;

- The density based algorithm implies the use of a parameter: the distance to which two points are considered part of the same cluster. Theoretically this could work, however, in the real world, different streets and venues have different characteristics, for example, some avenues have very specific drop-off

points for taxis while in others cases, for events in big buildings or gardens it's usual to see taxis stop all around the venue. This would force us to somehow adapt this parameters dynamically for each zone, to avoid, for example covering the whole street with just one cluster, or having simply too much clusters for a big area, which would happen if we were to assume all the areas to have a similar behavior.

The next step was to apply k-means, a very simple algorithm, where the number of clusters has to be selected a priori, independently of the data characteristics. This algorithm brings advantages and disadvantages with it. It can be general enough to extract a lot of potential POI's but it can also fail to identify the main point, for various reasons. This point can be too far away from the street, or the clustering algorithm could just perform badly on the specific pattern of the street.

The affinity propagation has shown visually some good results, but, like the others, it also had some examples where its performance would be inadequate. We decided not to used this clustering algorithm as its complexity is quadratic with the number of points and it would not be uncommon for an event to have more than 6,000 points, more than 3.6 million.

Finally I have developed a custom algorithm for this problem. This algorithm would extract the percentile 25 of the interpoint distances and then, for each point, calculate how many points where at a distance bellow this value. The point with the highest number of close points would then be selected as the Point of Interest.

Having now different ways to select the points of interest, we would still need to extract meaning from this coordinates. To do this we needed an API to return names for the Points for Interest. There are a number of options to do this, Foursquare has an API, and so does Instagram and other websites like Factual; Google does have this information but not a public API to access it. We also realized that when looking at our dashboard, that there were already some Points of Interest already marked, so we investigated and found out that OpenStreetMaps made use of an API when searching for names, called Nominatim. Because of this, we expect the results of the API to be in agreement with our maps.

When using an algorithm such as the k-means we do not get only one pair of coordinates but k different points instead. Nominatim would then be queried for these points, a typical response would include the street name, neighborhood, county and of course city and country. Some points had other specific characteristics, such as parking if there was a car park nearby. The information would then be collected for the k points and merged in such a way so there was no repetition and no conflicting information was lost. For a set of three points we could get the common fields, including different values for each one, if that is the case, and the other specific characteristics.

5.7.2 Date and time

As for date and time, we experimented with different formats and variations. For example, the same date could be written in various ways: "August 12 2012", "12th August 2012" or a more numerical approach "2012-08-12". The system would then create from a date, different possible ways of writing it. The system was even capable of detecting if the time of an outlier was either in the morning, afternoon, evening or even a full time event but after analyzing the resulting queries manually,

we realized there was a negative impact on the number of results when adding words like "morning" to the query, so we did not use this in the final system.

Another thing to take account to was that some events could be announced as being on the previous day at night but in fact, the outliers could only be visible on the following day. So the previous date and its written variations were also added to the possibilities.

Finally, we created a list of Singapore holidays and we would see if a certain date was one of them. If it was, it too would be added to the list of possible dates.

5.7.3 Results

In the end we would have a large combination of dates and times for each anomaly.

For example, Figure 5.6 shows an image of the automatic system, using k-means as the clustering algorithm. In this specific case the use of Nominatim API resulted in 6 different points of interest[4], while our date system returns at least two dates, if using only a single format. From the product of possibilities would result 12 queries, if one of theses dates was an holiday, this number would jump to 18. We decided to test different combinations and to see if we could remove some of these queries, knowing that Google does automatic query expansion and can even change the weight of some words, this should be possible.

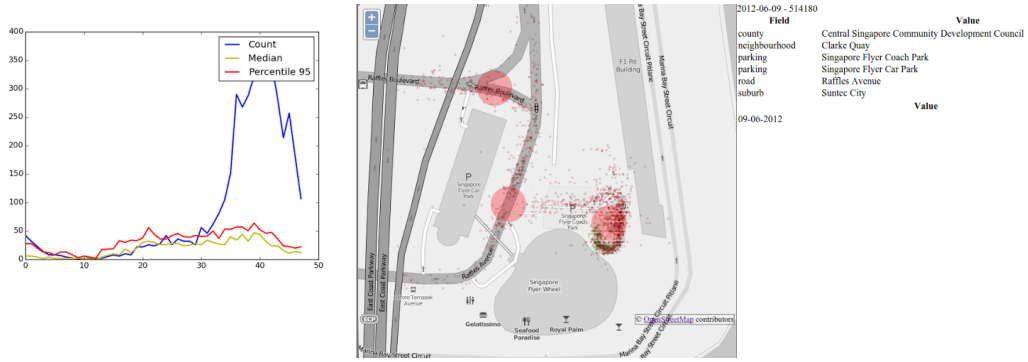


Figure 5.6: Improved version of the dashboard with K-means and Points of Interest.

To benchmark the system we would extract the domain names of the search results and evaluated its precision and recall against the domains that we found during the manual event identification, where we found a mean of *5.52 websites*, explaining each event. Table 5.1 shows these results. For the sake of simplicity, we call "1st format" for dates following the order YYYY-mm-dd, eg. "2012-08-12"; and we call "2nd format" dates written in full, like "8th August +2012".

[4]City and Country are removed.

POI	Date Format	Holiday	Days	Matches Mean	Found Links Mean	Precision	Recall
Manual	1st Format	No	Outlier Day	1.94736842105	9.94736842105	0.1958	0.3524
Manual	1st Format	No	Day Before	0	1.57894736842	**0**	**0**
Manual	2nd Format	No	Outlier Day	0.157894736842	0.157894736842	1.0000	0.0286
Manual	(1st Format) | (2nd Format)	No	Outlier Day	1.94736842105	9.94736842105	0.1958	0.3524
Manual	(1st Format) | (2nd Format)	Yes	Outlier Day	2.0	10.1578947368	0.1969	0.3619
3-means	(1st Format) | (2nd Format)	Yes	Outlier Day	0.263157894737	1.57894736842	0.1667	0.0476
Custom Algorithm	(1st Format) | (2nd Format)	Yes	Outlier Day	0.263157894737	1.0	0.2632	0.0476

Table 5.1: Dataset with number of hits and number of words for each query.

One of the first conclusions to take from the tests was that using the previous days didn't actually result in better values, there are a couple of possible explanations for this. We expected to capture two different kinds of events using this technique, the ones related to nightlife where people would get to the event after midnight and events that are celebrated at midnight. What we found was that nightlife in Singapore usually starts before midnight, usually at 23 or sooner, and that for events that are celebrated at midnight, even thought some outliers can exist on the previous day (usually present in both) these events are usually announced with the name of the holiday, and not the day preceding it. We decided to remove the option of using the day before on the following tests for these same reasons.

By using the second date format, we no longer receive as many links as expected. However, every link returned was about the event we wanted identify. Another thing about dates is that we can conclude, after using both date formats, that the second format was already used by Google when using the first one, this is a good example of *query expansion*. Because of this, there is no improvement or deterioration on the system's performance. Finally we created a list of holidayswhich we then used [5] on the creating of new queries. This made the performance of the algorithm rise both in precision and recall.

We then tested different ways of selecting the Points of Interest, both using K-means and the custom algorithm described before in Subsection5.7.1. These now represent a fully automated system. These results were expected not be as good as a manual system, even though their performance was slightly better than other existing manual systems. It is worth noticing that these systems found very little links, with a mean of about 1.57 and 1 for K-means and the custom algorithm respectively. While the k-means precision was not better of any of the other tested setups, which was expected for a system that always chooses 3 points, making assumptions about the points geometry. The custom algorithm helped the system to identify the Point of Interest better than the manual identification did, this is likely because Nominatim does not only return just one POI name but a list of possibilities, one per field, at least.

Finally we ran the last tests using only the top 1 and top 3 results from Google, Table 5.2 shows a comparison between using these or all the 10 links return in the first page.

As we can see from these results, doing this would result in a trade-of between precision and recall. It was clear that doing this would reduce the mean number

[5]Holidays used: New Years Day, Chinese New Year, Good Friday, Labour Day, Vesak Day, National Day, Hari Raya Haji, Deepavali, Christmas Day, Ramadan.

POI	Date Format	Top	Matches Mean	Found Links Mean	Precision	Recall
Manual	(1st Format) | (2nd Format)	10	2.0	10.1578947368	0.1969	0.3619
Manual	(1st Format) | (2nd Format)	3	1.10526315789	2.84210526316	0.3889	0.2000
Manual	(1st Format) | (2nd Format)	1	0.526315789474	0.95789473684	0.5494	0.0952
3-means	(1st Format) | (2nd Format)	10	0.263157894737	1.57894736842	0.1667	0.0476
3-means	(1st Format) | (2nd Format)	3	0.210526315789	0.736842105263	0.2857	0.0381
3-means	(1st Format) | (2nd Format)	1	0.210526315789	0.421052631579	0.5000	0.0381
Custom Algorithm	(1st Format) | (2nd Format)	10	0.263157894737	1.0	0.2632	0.0476
Custom Algorithm	(1st Format) | (2nd Format)	3	0.210526315789	0.473684210526	0.4444	0.0381
Custom Algorithm	(1st Format) | (2nd Format)	1	0.210526315789	0.315789473684	0.6667	0.0381

Table 5.2: Dataset with number of hits and number of words for each query.

of results but the higher precision would probably be the better option for a real application.

5.8 Discussion and Improvements

Some events were almost impossible to identify, for example, there was an outlier with more than 200 drop-offs close to a military facility. This was extremely hard to identify and we were only able to understand what happened in that place after looking up military events in Singapore. Only then, we knew that was one of the cadet's graduation days, after identify a single blog post, from a soldier, talking about the event itself. Some events were even more bizarre: there was a large number of drop-offs late at night in a very specific urban neighborhood, with apparently not a single point of interest nearby. Stranger yet, most of these people have come from a very specific place, another, seemingly, very uninteresting place. We were not able to identify anything related to this event.

It was also interesting to see some outliers on some holidays at the Singapore power station and we could even identify some changes of shifts and other curious patterns. One of the most interesting ones was a very high number of drop-offs close to the police academy, just one day before Singapore's National Day, probably the last mission briefing before the big day.

We think that even though this experiment was successful, there is room for improvement. The first would be for example the development of a classifier that would go through the search results and identify which one would most likely identify the event. This would be particularly important when websites such as Wikipedia, are included in the search results, that, even though may bring some relevant importance to the specific venue or holiday, will most likely not identify the specific event we are trying to find. There have been similar approaches to this, which are described in Section s2.3. Another improvement, would be to make the system use Event Aggregation websites API such as All Events In `allevents.in` and look directly into this websites. However, this would create more dependencies for the system and make the general context sensing from the web only a secondary feature.

Another thing to keep in mind is that the manual identification of events is not perfect, in fact, there is a strong possibility that we did not capture all the relevant links and seeing that the automated system using the custom algorithm for POI identification had some better results than the system using the manual identified POI's of interest, points us in that direction. We suspect that the system was capable, using the Nominatim API to find better terms to look up the venue and indirectly get more relevant results than some of our queries.

Chapter 6

Conclusion

This dissertation studied the application of different techniques with the shared objective of selecting automatically what would be the most informative information to use. The foremost application comes in a form of an environmental sensing system. This system would have one movable sensor only, and the possibility to change it between a limited number of places. We developed a strategy that applies an Active Sensing setup to choose which sensor local to choose next, i.e. which information to feed next to a regression model. This strategy turned out to be better than choosing the sensors with the most available information and the random, uniform, selection of locals. The second application consisted of predicting the number of results a search engine would return for a given query. We focused on predicting queries related to events happening in Singapore. We have learned how hard this task was for our model since the results were never satisfactory. Our goal then evolved to be the identification of traffic anomalies in Singapore and also the identification of which events may have caused it. Starting with a manual search, we went through an iterative process of automatizing this search and identification of events. In order to do different experiments, we could not get into much detail for each problem. We are sure that both of them can be improved, if there is sufficient motivation to do so, which we think is the case.

Even though the dissertation experiments were mostly successful, we are confident that having different datasets for both Chapter 3 and Chapter 4 would have brought us better end results. In the first case, having a better quality dataset, would most likely, reduce much of the needed pre-processing and make it easier to model using Gaussian Processes; for the second case, we suspect that 4,829 would be an insufficient number of queries to analyze, for a problem of such high complexity. Fortunately, the last dataset, even though challenging to process due to its size, did not share this problem of invalid data.

This dissertation shows different contexts where parsimonious sensing can be applied, and how different techniques can be used to optimize the process of reducing the quantity of information necessary for a system to compute. We believe parsimonious sensing can allow for needed efficient selection of information to study, in the following years, as the amount of documents grows.

Appendix A

Kernel Functions

Some papers and books define the same kernels differently, some with more, others with less arguments, this chapter presents the most commented kernels, both in [53] and [15]. This appendix is to be used as a visual aid to comprehend the behavior of kernel functions and their formulas.

Each of the functions is presented with its formula followed by two samples from the prior. The arguments values used were chosen only for visualization purposes and are ,probably, not the best, performance-wise.

Section A.1 shows the behavior of a covariance function with different lenght-scales, this can serve as an example for other functions.

A.1 Squared Exponential

The de-facto default kernel for GPs and SVMs, also known as *RBF* or Radial Basis Function.

It works very well with smooth, continuous functions, but it can fail to model real data.

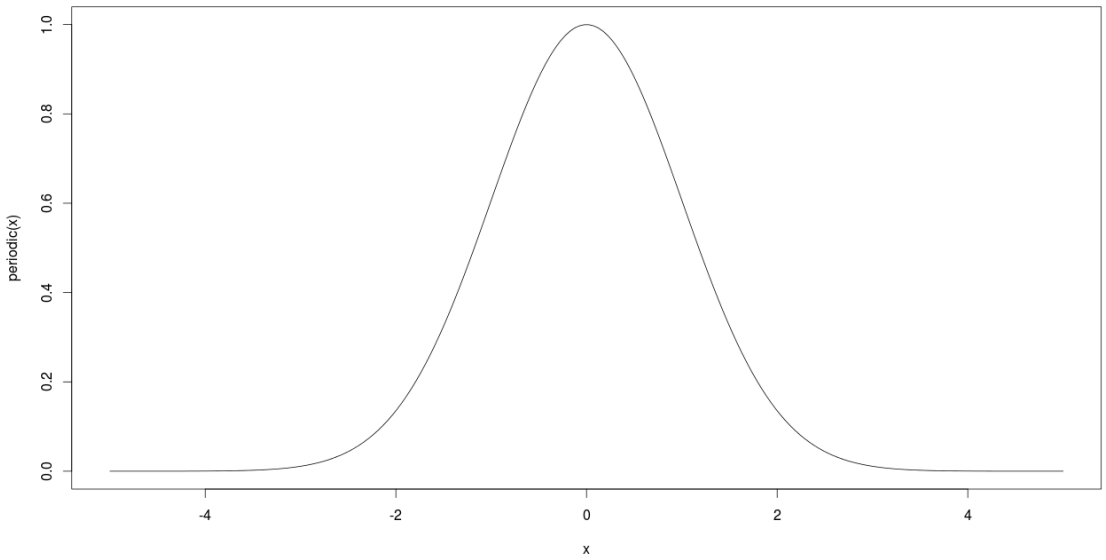


Figure A.1: Squared Exponential Function, $\ell = 1$.

$$k_{\mathrm{SE}}(x, x') = \sigma^2 \exp\left(-\frac{||x - x'||^2}{2\ell^2}\right) \tag{A.1}$$

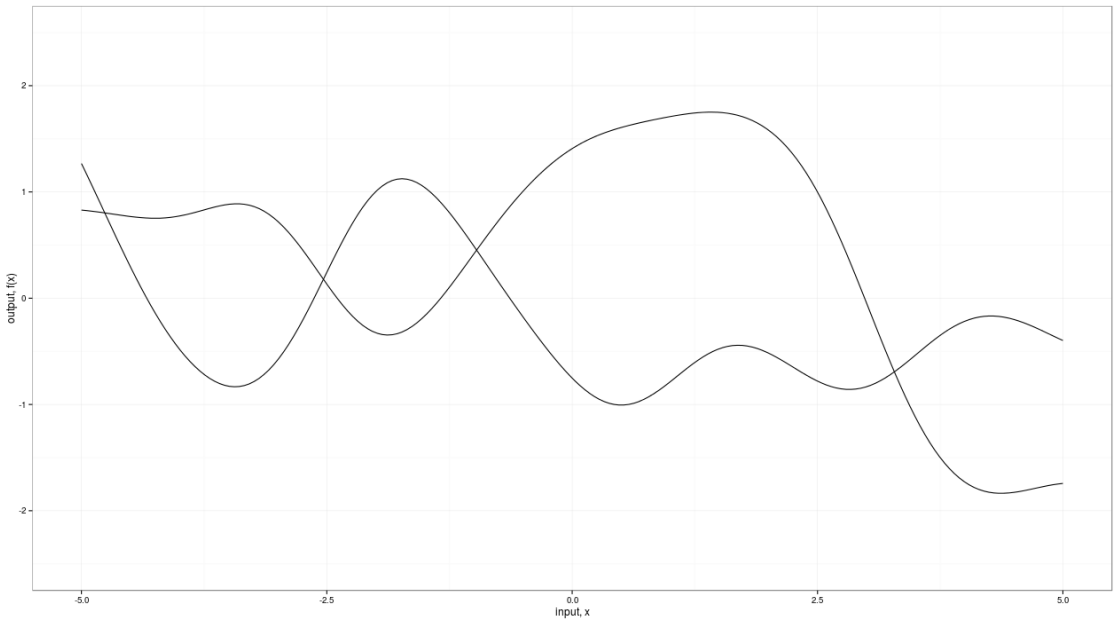


Figure A.2: Squared Exponential Samples, $\ell = 1$.

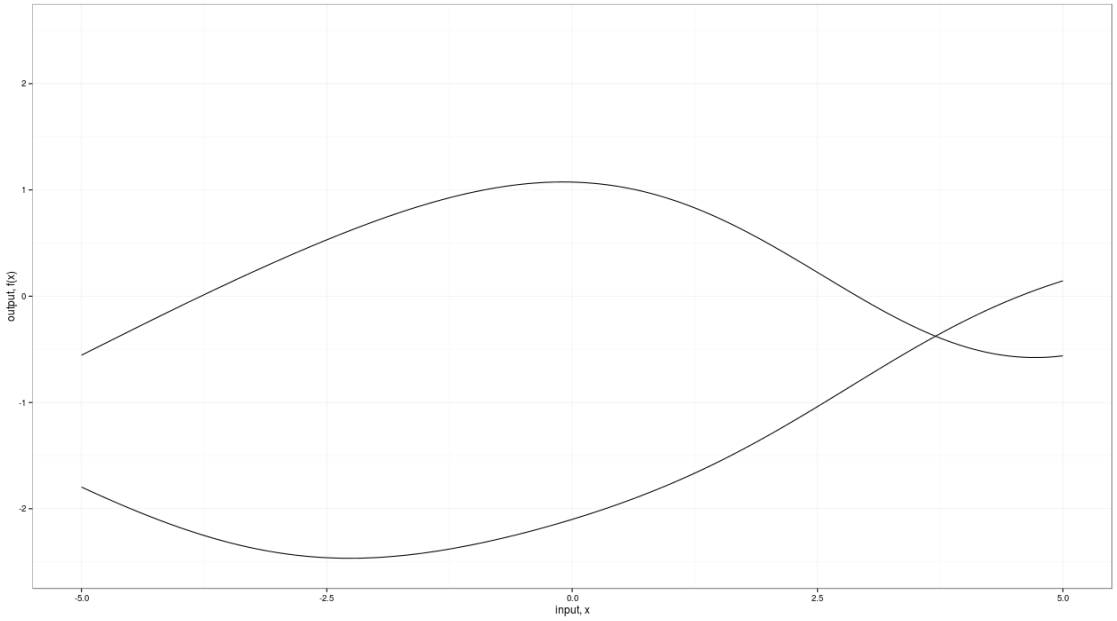


Figure A.3: Squared Exponential Samples, $\ell = 3$.

A.2 Rational Quadratic Kernel

This kernel is equivalent of many Squared Exponential Kernels with different length-scales added.

When $\alpha \to \infty$ it is identical to SE. It also shares the problem of beeing too smooth for most data.

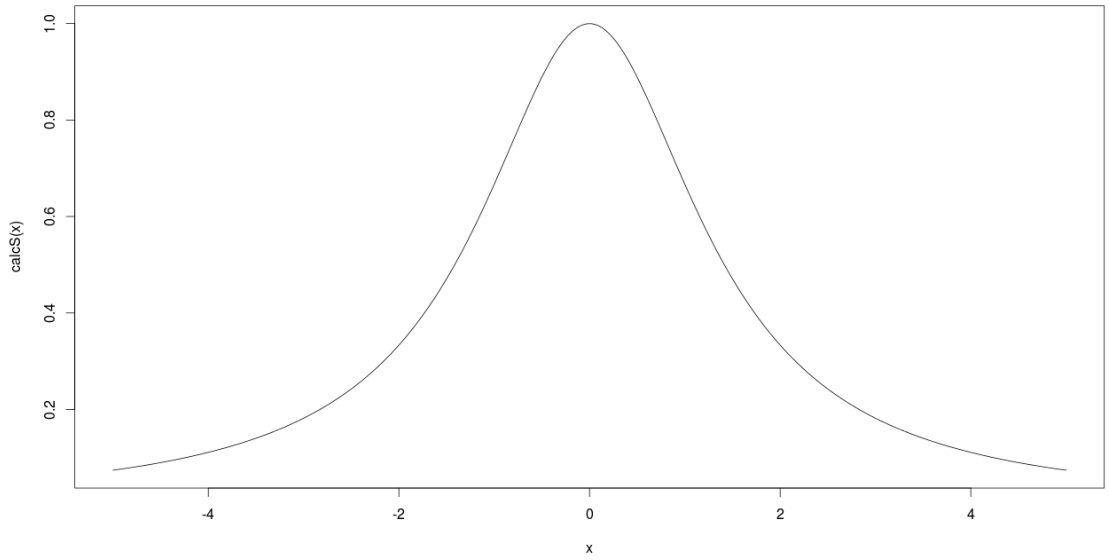


Figure A.4: Rational Quadratic Function, $\ell = 1$, $\alpha = 2$.

$$k_{\mathrm{RQ}}(x, x') = \sigma^2 \left(1 + \frac{(x - x')^2}{2\alpha\ell^2}\right)^{-\alpha} \tag{A.2}$$

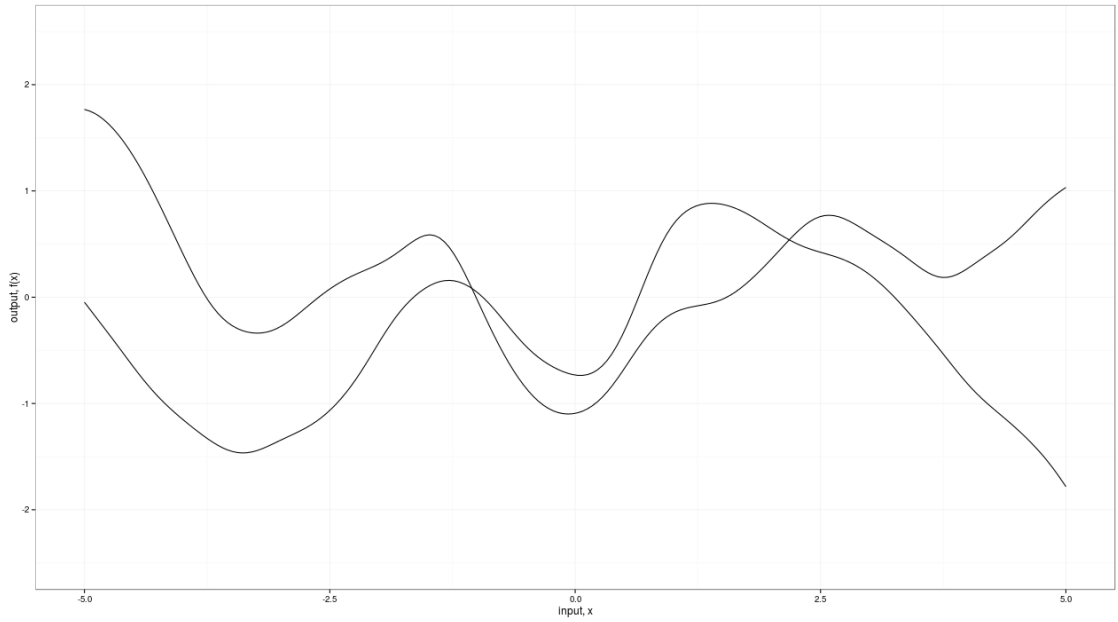


Figure A.5: Rational Quadratic Kernel $\ell = 1$, $\alpha = 2$.

A.3 Periodic Kernel

It allows the model to make predictions over functions which repeat themselves.

p is the period, the distance between repetitions.

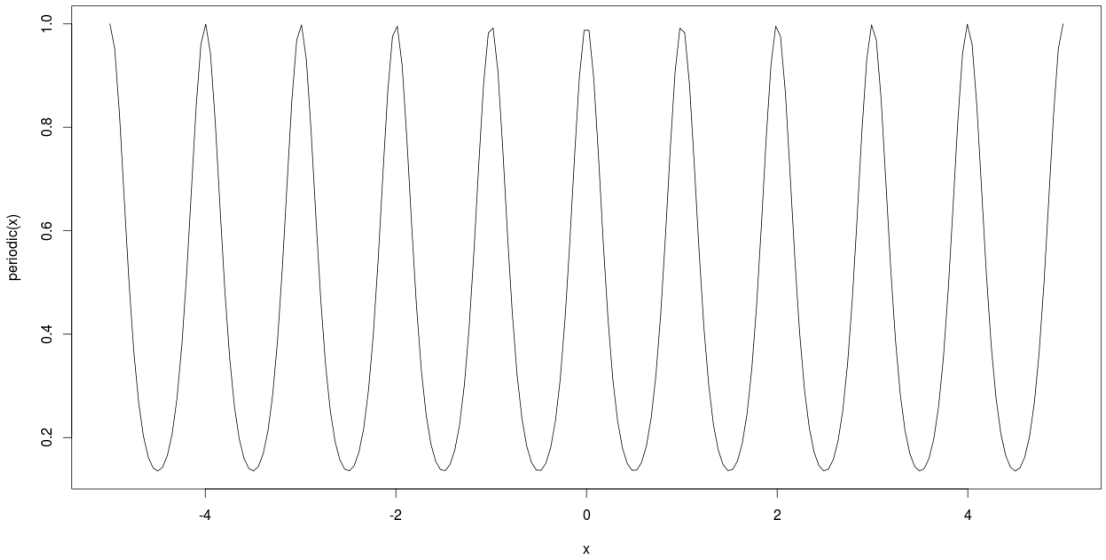


Figure A.6: Periodic Function, $\ell = 1$ and $p = 1$.

$$k_{\text{Per}}(x, x') = \sigma^2 \exp\left(-\frac{2\sin^2(\pi||x - x'||/p)}{\ell^2}\right) \tag{A.3}$$

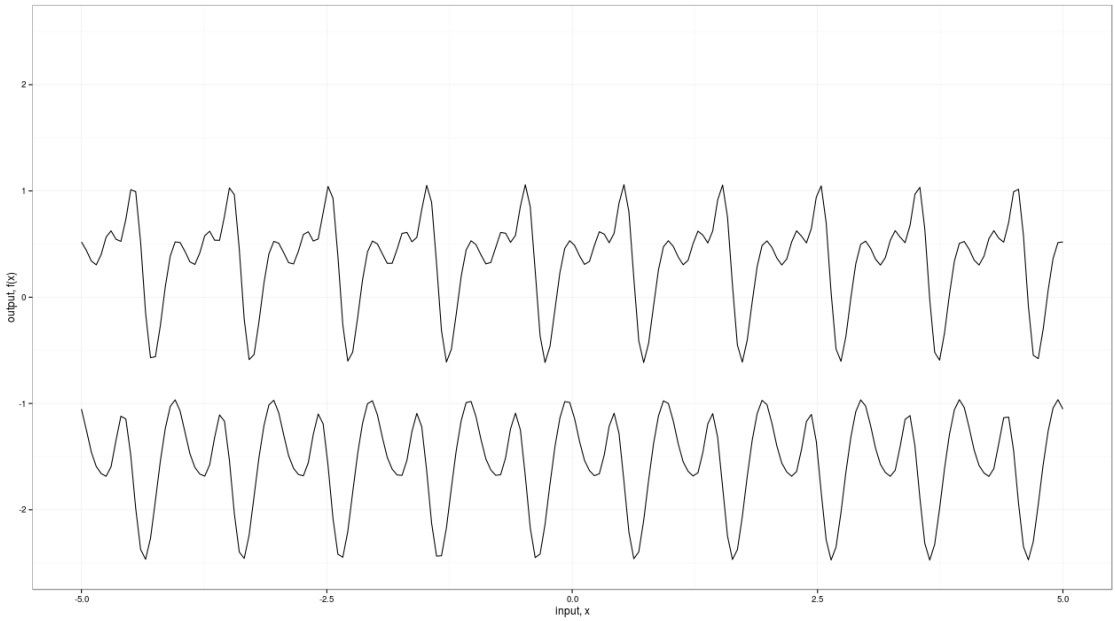


Figure A.7: Periodic Kernel, $\ell = 1$ and $p = 1$.

A.4 Locally Periodic Kernel

A function capable of modeling functions which are periodic but that change over time results from multiplying a SE kernel with a Periodic Kernel. Because most

periodic functions do not repeat themself exactly, this adds more flexibility to the model.

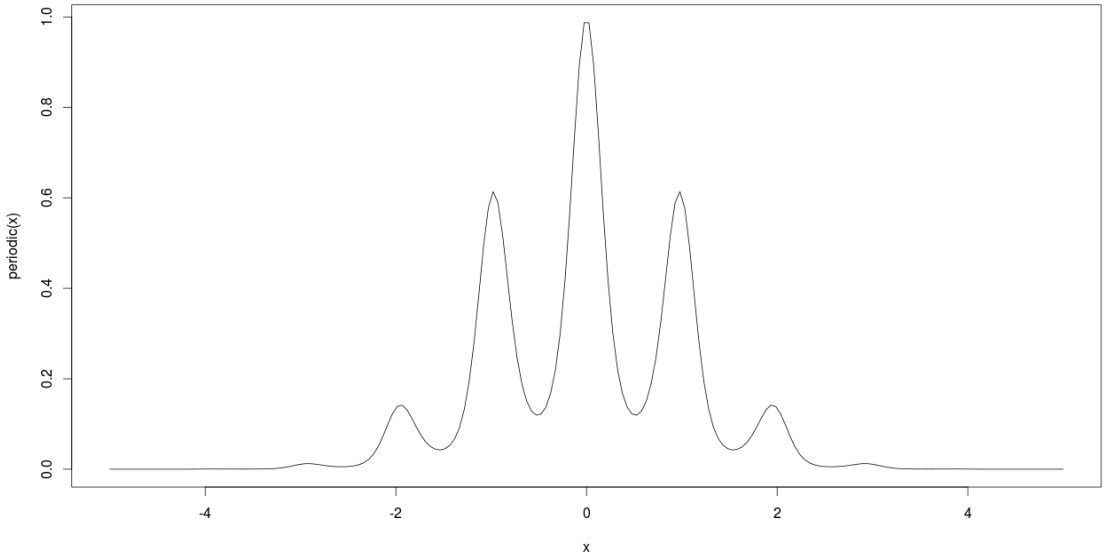


Figure A.8: Locally Periodic Function, $\ell = 1$ and $p = 1$.

$$k_{\text{LocalPer}}(x, x') = k_{\text{Per}}(x, x') k_{\text{SE}}(x, x') \tag{A.4}$$

$$= \sigma^2 \exp\left(-\frac{2\sin^2(\pi||x - x'||/p)}{\ell^2}\right) \exp\left(-\frac{||x - x'||^2}{2\ell^2}\right) \tag{A.5}$$

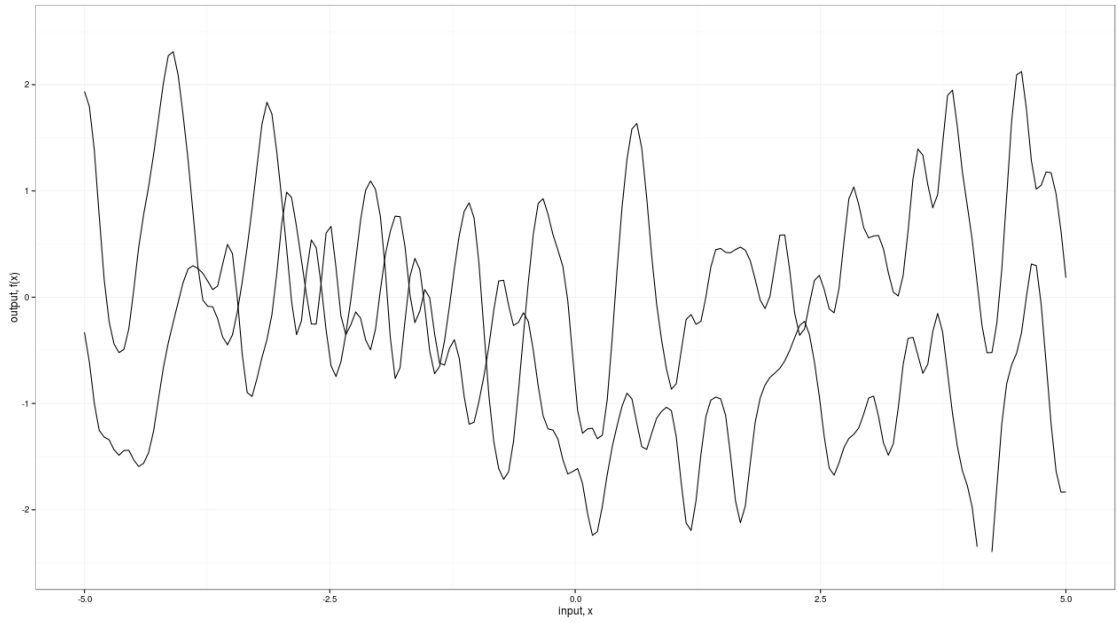


Figure A.9: Locally Periodic Kernel, $\ell = 1$ and $p = 1$.

Appendix B

Extra LDA Analysis

This Appendix shows a visualization of the topics distributions of the queries using LDA, for different number of topics. Multi-dimensional scaling is used for the dimensionality reduction. Figures B.1, B.2, B.3, B.4, and B.5, show the different visualizations for 3, 4, 8, 20 and 40 topics, respectively.

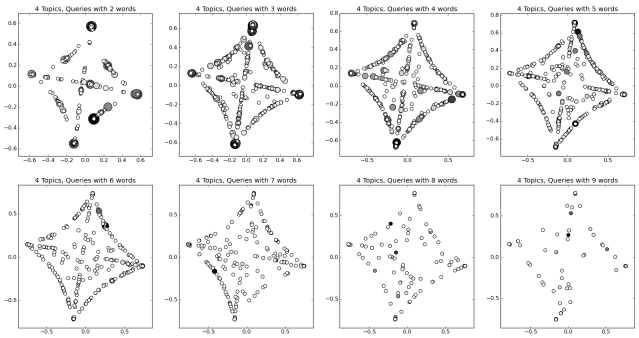


Figure B.1: Distribution for the number of results, 3 topics.

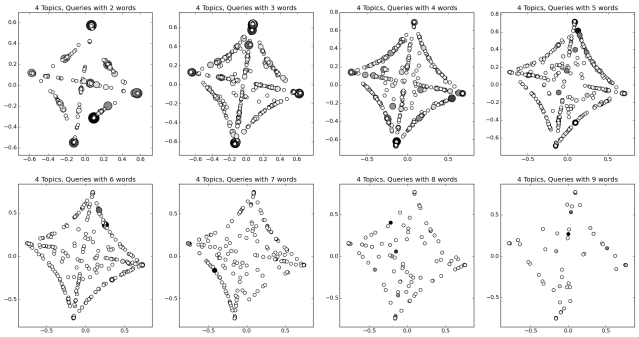


Figure B.2: Distribution for the number of results, 4 topics.

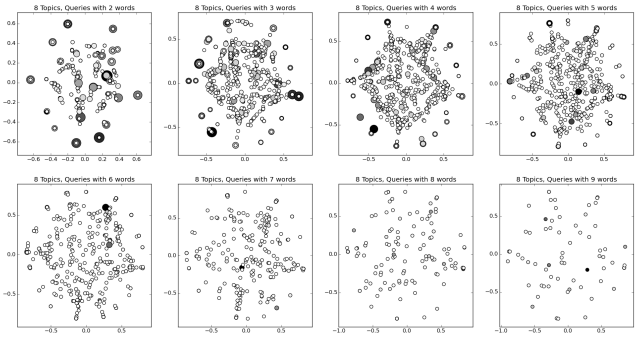


Figure B.3: Distribution for the number of results, 8 topics.

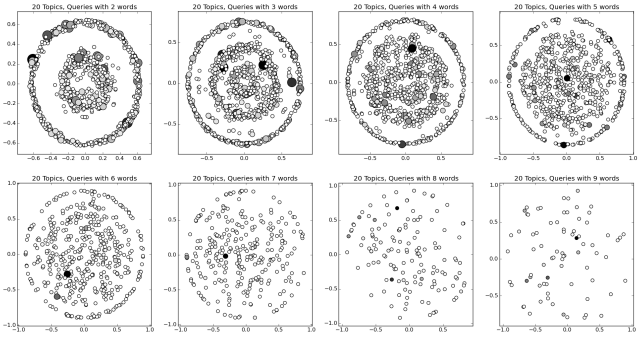


Figure B.4: Distribution for the number of results, 20 topics.

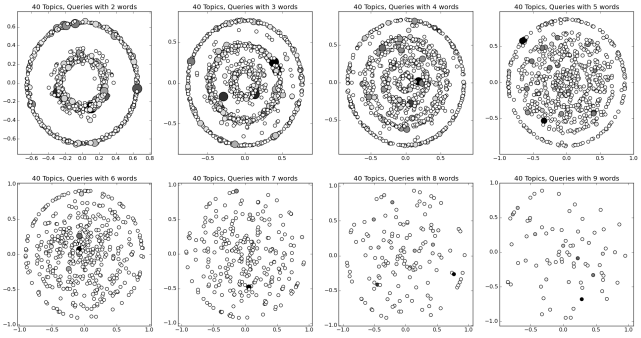


Figure B.5: Distribution for the number of results, 40 topics.

Appendix C

Python, R and Matlab interaction

During the development of this thesis I explored the application of different programming languages to the problem of context and parsimonious sensing. Each of these languages have their own advantages over the others. R allowed for very easily describing, both visually and numerically the datasets and also their transformations, like normalization and filtering. Matlab has, what I have found to be, personally, the best Gaussian Processes library. Complete with many different likelihood, covariance functions, it allowed for a freedom that most other libraries did not. Its code is also very optimized, while using Python's sklearn library to do regression using Gaussian Processes, invariably resulted in IO trashing because of the memory it required while inverting matrices. While Python, being one of the languages which I am more comfortable writing allowed me to write new easily readable tests very quickly. Another reason to use Python was the library Hyper-opt, which was essential for finding good hyper-parameters for the Gaussian Processes.

Easing these three languages separately made me have to develop big script files that would run programs wrote in different languages and each of them being feed the input of the previous one, this was obviously not good, and most times a simple modification of a test would be extremely hard to debug, which made me lost valuable time.

I came to the conclusion that the perfect solution would be to have good libraries for the language which I am most fluent in, Python and try to develop the tests using only this. After some research I found Pandas, a data analysis library for Python which allowed me to do operations in a similar way as one would in R, which was great, but did not have the same visualization capabilities as R does, natively. I found however that the library I was using for most of my visualizations in R called *ggplot2* was being ported to Python, and even though there were still some bugs in this implementation, I found that my specific application did not suffer much from this, and in fact, only a slight modification of the datasets would allow me to use it in a very similar way as I was using it before in R. Finally, the big challenge was to find a complete Gaussian Processes library that would not be as limited as sklearn in terms of option and that would also make good management of the system's memory and CPU. I could not find this, instead, I thought of developing a module that would establish a connection between matlab and Python, but only the last version of matlab supported this connection, which I did not have access to. I was running out of options until I remembered that GPML is compatible with octave, an open-source matlab alternative, that although is not as good performance wise,

should be enough. The result was an experimental Python class that would receive Gaussian Processes hyper-parameters and would have the methods fit and predict. After developing this class, I ported most of the tests that I would, most likely, alter and re-run to Python, which made the process easier and cleaner.

Appendix D

Working with Big Data

Working with large datasets proved to be a problem.

As most of the scripts would take a long time to run, most of the times I implemented a new one, I would first test it on a small sample, limiting the rows read by the script.

Applying this technique would enable me to perform many more experiences than I would if running over the complete 54 GB dataset. However, it also brought some unexpected problems from which I learned.

D.1 Reading large files with Python

By default, pandas[1], when opening a dataset from a CSV[2] file, tries to guess the column types. This process hogs up memory, provoking IO trashing very quickly in a 16 GB's machine, for a relatively small file of 1 GB. The first time I solved this problem, I did it by explicitaly defining the columns types, however this was insuficient for some later analysis.

D.2 Processing chunks of information

For the longer tests, I would have the script to print every few seconds the expected time to finish.

Extracting statistics for specific zones, dates and times of the day, the tests was expected to take at least two weeks to run, on a single thread[3]. For this test, there was a CSV file with the statistics to run, besides some other information, each line would contain a zone and date. The script would then lookup a database for more information about that zone in the dates that would correspond to the same week day. I immediately tried to tap into the other threads of the CPU, by trying to make scripts that would make use of multi-core enabled libraries or python 3 multiprocessing library. After all, most of these tests were, what is considered "embarrassingly parallel", i.e. it would seem that for little effort, one could separate the problem into a number of parallel tasks.

[1] A data analysis library with similar structures as R.

[2] Comma Separated File

[3] The used setup had a Intel i7-4700HQ Processor, 64 bits, 4 cores, 8 threads, 2.4 GHz to 3.4 GHz, 6 MB of cache with 16 GB of RAM.

First, I have developed the script using a sample of the data and I would benchmark them against the single threaded version. Once I achieved reasonable results I would try to run it on the complete dataset. However, when doing this, one must remember that the multiprocessing library is based on the fork system call, and, because of this, it creates copies of the current process, multiplying the used memory by, at least, the number of processes opened. Using shared memory was not an option since the library API only provides very limited data structures for it.

The next experiment would be to read the files line by line and send each line for a different process, this proved efficient for the small dataset, however, for large datasets it became slower then using a single thread. Finally I understood that reading the file line by line, even though was memory efficient, it was not an efficient way to read the files and it lost performance over the process management overhead. The best and final solution would be to read thousands of lines each time, fill an array with them and then send them to each separated process.

Adding a semaphore to the script, so only one thread would lookup the database at a time resulted in some added performance.

Finally, using 7 of the 8 available CPU threads, with the semaphore, some memory optimization and the creation of database indexes for the columns "zones" and "date" would ultimately reduce the running time of some tests from two weeks to little more than one and a half day, almost 10 times faster.

Unfortunately, this was not the case for every test. It was not possible to run more than two, sometimes, three, processes using Gaussian Processes regression since they would quickly exhaust any available system memory causing the system to either crash or trash the hard disk.

Bibliography

[1] Hamed Abdelhaq, Christian Sengstock, and Michael Gertz. Eventweet: Online localized event detection from twitter. *Proceedings of the VLDB Endowment*, 6(12):1326–1329, 2013.

[2] Gregory D Abowd, Anind K Dey, Peter J Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *Handheld and ubiquitous computing*, pages 304–307. Springer, 1999.

[3] Elizabeth Bales, Nima Nikzad, Nichole Quick, Celal Ziftci, Kevin Patrick, and William Griswold. Citisense: Mobile air quality sensing for individuals and communities design and deployment of the citisense mobile air-quality system. In *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2012 6th International Conference on*, pages 155–158. IEEE, 2012.

[4] Eric B Baum and Kenneth Lang. Query learning can work poorly when a human oracle is used. In *International Joint Conference on Neural Networks*, volume 8, 1992.

[5] Hila Becker, Mor Naaman, and Luis Gravano. Learning similarity metrics for event identification in social media. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 291–300. ACM, 2010.

[6] Hila Becker, Mor Naaman, and Luis Gravano. Beyond trending topics: Real-world event identification on twitter. *ICWSM*, 11:438–441, 2011.

[7] Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[8] Sergey Brin and Lawrence Page. Reprint of: The anatomy of a large-scale hypertextual web search engine. *Computer networks*, 56(18):3825–3833, 2012.

[9] Robert Burbidge, Jem J Rowland, and Ross D King. Active learning for regression based on query by committee. In *Intelligent Data Engineering and Automated Learning-IDEAL 2007*, pages 209–218. Springer, 2007.

[10] Michelle Nicole Burns, Mark Begale, Jennifer Duffecy, Darren Gergle, Chris J Karr, Emily Giangrande, and David C Mohr. Harnessing context sensing to develop a mobile intervention for depression. *Journal of medical Internet research*, 13(3), 2011.

[11] Francesco Calabrese, Giusy Di Lorenzo, Gavin McArdle, Fabio Pinelli, and Erik Van Lierde. Real-time social event analytics, 2015.

[12] Aron Culotta and Andrew McCallum. Reducing labeling effort for structured prediction tasks. In *AAAI*, pages 746–751, 2005.

[13] Anind K Dey. Understanding and using context. *Personal and ubiquitous computing*, 5(1):4–7, 2001.

[14] Anind K Dey, Gregory D Abowd, et al. The context toolkit: Aiding the development of context-aware applications. In *Workshop on Software Engineering for wearable and pervasive computing*, pages 431–441, 2000.

[15] David Kristjanson Duvenaud. *Automatic Model Construction with Gaussian Processes*. PhD thesis, University of Cambridge, 2014.

[16] Alois Ferscha, Simon Vogl, and Wolfgang Beer. Ubiquitous context sensing in wireless environments. In *Distributed and Parallel Systems*, pages 98–106. Springer, 2002.

[17] John Foley, Michael Bendersky, and Vanja Josifovski. Learning to extract local events from the web. 2015.

[18] Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58, 1992.

[19] Marta C Gonzalez, Cesar A Hidalgo, and Albert-Laszlo Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, 2008.

[20] Kenneth P Hinckley and Eric J Horvitz. Mobile phone operation based upon context sensing, November 27 2007. US Patent 7,302,280.

[21] Alex Holub, Pietro Perona, and Michael C Burl. Entropy-based active learning for object recognition. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*, pages 1–8. IEEE, 2008.

[22] Sheng-Jun Huang, Rong Jin, and Zhi-Hua Zhou. Active learning by querying informative and representative examples. In *Advances in neural information processing systems*, pages 892–900, 2010.

[23] Rebecca Hwa. Sample selection for statistical parsing. *Computational Linguistics*, 30(3):253–276, 2004.

[24] Erik Thorlund Jepsen, Piet Seiden, Peter Ingwersen, Lennart Björneborn, and Pia Borlund. Characteristics of scientific web publications: Preliminary data gathering and analysis. *Journal of the American Society for Information Science and Technology*, 55(14):1239–1249, 2004.

[25] Ajay J Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. Multi-class active learning for image classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2372–2379. IEEE, 2009.

[26] Adam Kilgarriff and Gregory Grefenstette. Web as corpus. In *Proceedings of Corpus Linguistics 2001*, pages 342–344. Corpus Linguistics. Readings in a Widening Discipline, 2001.

[27] Soohwan Kim and Jonghyuk Kim. Gpmap: A unified framework for robotic mapping based on sparse gaussian processes. In *Field and Service Robotics, Results of the 9th International Conference*. Springer, 2015.

[28] Christine Körner and Stefan Wrobel. Multi-class ensemble-based active learning. In *Machine Learning: ECML 2006*, pages 687–694. Springer, 2006.

[29] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, pages 591–600. ACM, 2010.

[30] David D Lewis and Jason Catlett. Heterogenous uncertainty sampling for supervised learning. In *ICML*, volume 94, pages 148–156, 1994.

[31] David D Lewis and William A Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–12. Springer-Verlag New York, Inc., 1994.

[32] Bing Liu. *Web data mining: exploring hyperlinks, contents, and usage data*. Springer Science & Business Media, 2007.

[33] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[34] Mor Naaman, Jeffrey Boase, and Chih-Hui Lai. Is it really about me?: message content in social awareness streams. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, pages 189–192. ACM, 2010.

[35] Preslav Nakov and Marti Hearst. A study of using search engine page hits as a proxy for n-gram frequencies. In *Proceedings of RANLP*, volume 5. Citeseer, 2005.

[36] Nicholas Roy and Andrew McCallum. Toward optimal active learning through monte carlo estimation of error reduction. *ICML, Williamstown*, 2001.

[37] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM, 2010.

[38] Daniel Salber, Anind K Dey, and Gregory D Abowd. The context toolkit: aiding the development of context-enabled applications. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 434–441. ACM, 1999.

[39] Daniel Salber, Anind K Dey, Robert J Orr, and Gregory D Abowd. Designing for ubiquitous computing: A case study in context sensing. 1999.

[40] Jagan Sankaranarayanan, Hanan Samet, Benjamin E Teitler, Michael D Lieberman, and Jon Sperling. Twitterstand: news in tweets. In *Proceedings of the 17th acm sigspatial international conference on advances in geographic information systems*, pages 42–51. ACM, 2009.

[41] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In *Advances in Intelligent Data Analysis*, pages 309–318. Springer, 2001.

[42] Andrew I Schein and Lyle H Ungar. Active learning for logistic regression: an evaluation. *Machine Learning*, 68(3):235–265, 2007.

[43] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52:55–66, 2010.

[44] Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1070–1079. Association for Computational Linguistics, 2008.

[45] Burr Settles, Mark Craven, and Lewis Friedland. Active learning with real annotation costs. In *Proceedings of the NIPS Workshop on Cost-Sensitive Learning*, pages 1–10, 2008.

[46] Burr Settles, Mark Craven, and Soumya Ray. Multiple-instance active learning. In *Advances in neural information processing systems*, pages 1289–1296, 2008.

[47] H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294. ACM, 1992.

[48] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[49] Michael L Stein. *Interpolation of spatial data: some theory for kriging*. Springer, 1999.

[50] Hideyuki Takagi. Interactive evolutionary computation: Fusion of the capabilities of ec optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296, 2001.

[51] Ondřej Veselỳ and Jan Kolomazník. Predicting number of search engine results to optimise online plagiarism detection. *Proceedings of Plagiarism across Europe and Beyond*, 2013.

[52] Andreas Weiler, Marc H Scholl, Franz Wanner, and Christian Rohrdantz. Event identification for local areas using social media streaming data. In *Proceedings of the ACM SIGMOD Workshop on Databases and Social Networks*, pages 1–6. ACM, 2013.

[53] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. 2006.

[54] Leo Yuen, Matthew Chang, Ying Kit Lai, and Chung Keung Poon. Excalibur: a personalized meta search engine. In *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International*, volume 2, pages 49–50. IEEE, 2004.