Imperial College London

Department of Computing

Individual Project: Final Report

Machine Learning for a London Housing Price Prediction Mobile Application

Author:
Aaron Ng

Supervisor:
Dr. Marc Deisenroth

Submitted in partial fulfilment of the requirements for BEng in Electronics and Information Engineering

June 2015

Abstract

In this project, we present a mobile application that can generate predictions for future housing prices in London. In order to select a prediction method, various regression methods were explored and compared. Gaussian Processes (GP) for regression was chosen as our model due to its flexible and probabilistic approach to learning and model selection. To handle the large dataset of past property transactions in London, we exploit the spatial structure of the dataset to distribute computations to smaller independent local models. Overall predictions are obtained by recombining predictions from local models. By training the model and generating predictions on the server-side of the application, we are able to offload computationally intensive tasks and focus on generating visualisations on the client-side. Our results demonstrate that our approach to the problem has been largely successful, and is able to produce predictions that are competitive to other housing price prediction models.

Acknowledgements

I would like to thank the following people for their support in this project:

- Dr Marc Deisenroth for his invaluable guidance and patient supervision throughout the entire project,
- the Computing Support Group at Imperial for advising me on the computational requirements of the project, and
- my family and friends for their unwavering love and support.

Contents

1 Introduction **3**

1.1 Objectives 4

2 Background **6**

2.1 London Housing Market 6

2.2 Dataset 7

2.3 Related Work 8

2.3.1 Property Agents and Economists 8

2.3.2 Property Search Websites 9

2.3.3 Summary 10

3 Regression Methods **11**

3.1 Introduction 11

3.2 Linear Regression 12

3.2.1 Error Function and Maximum Likelihood Solution 12

3.2.2 Regularisation 14

3.3 Bayesian Linear Regression 15

3.3.1 Posterior Distribution 15

3.3.2 Predictive Distribution 16

3.4 Relevance Vector Machines 16

3.4.1 Marginal Likelihood Maximisation 17

3.4.2 Predictive Distribution 18

3.4.3 Problems with Predictive Variances 19

3.5 Gaussian Process 20

3.5.1 Training 21

3.5.2 Predictive Distribution 22

3.5.3 Distributed GPs 22

3.5.3.1 Product of Experts 23

3.5.3.2 Generalised Product of Experts 23

3.6 Summary 24

4 Design and Implementation: Prediction System **25**
4.1 Data Preparation 25
4.1.1 Utilisation of Data 26
4.2 Training 27
4.2.1 Linear Prior Mean Function 27
4.2.2 Kernel 29
4.3 Prediction 30

5 Design and Implementation: Mobile Application **32**
5.1 Client-side 32
5.1.1 Application Structure 32
5.1.2 Application Design 34
5.2 Server-side 37
5.2.1 Web Server 37
5.2.2 Application Server 38

6 Evaluation **40**
6.1 Prediction Method 40
6.2 Prediction System 41
6.3 Mobile Application 42

7 Conclusion **43**
7.1 Future Work 43

Bibliography **45**

Chapter 1

Introduction

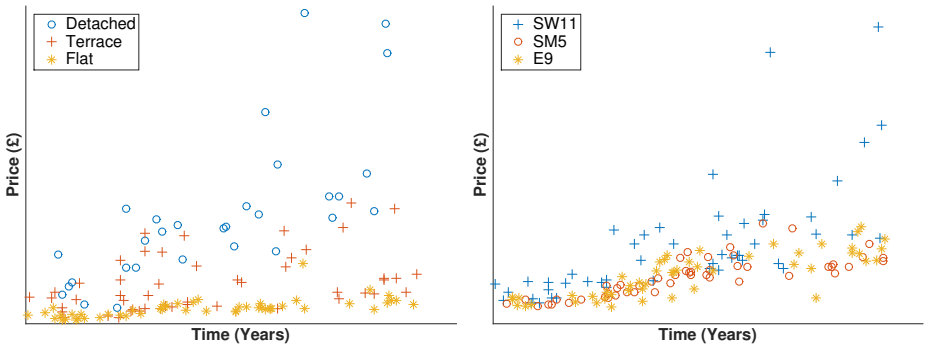
The housing market in the UK has always been a topic of national attention, with news sites dedicating sections that report on key news that can potentially affect housing prices and the trends in recent months. Not only are the trends in housing market of concern to buyers and owners, they reflect the current economic situation and social sentiments in the country.

For many people, buying a property is one of the most important decision and purchase in life. Besides the affordability of a house, other factors such as the desirability of the location and the long-term investment prospects also affect the decision-making process.

However, many reports [1, 2] on housing prices are averaged across regions in the UK, where London is considered one region. London homebuyers who want to find out more about housing trends in various districts would therefore have little recourse. For example, housing prices in Barking & Dagenham are likely to differ from those in Kensington & Chelsea, but it is difficult to find sources that provide such information. If there were location-specific predictions on housing prices within London, making informed decisions over which areas to consider can be greatly facilitated.

Data from the Land Registry[1] shows that there are over 2 million property transactions that have taken place in London since 1995. Making sense of this large dataset is by no means an easy feat. In Figure 1.1, we can see random samples of London property prices plotted against time. The graph on the left highlights the difference between property prices of different housing types, while the graph on the right highlights the difference between property prices taken from different areas. It is clear that many variables affect the trend in housing prices, which motivates the need for a prediction model that encompasses the relationships between these variables and housing prices.

[1] http://landregistry.data.gov.uk/



(a) Housing type (b) Location

Figure 1.1: Random samples of property transactions taken from the Land Registry, sorted by different categories.

This is where *machine learning* comes into play. Instead of using hard-coded parameters and static program instructions, the prediction system can learn from the dataset to teach itself to refine its parameters and make data-driven predictions. With a prediction model, we aim to assist property buyers in making predictions on future London property prices by harnessing the power of the large dataset already available.

1.1 Objectives

The goal of the project is to create a mobile application that can provide users with location-specific predictions and trends on London housing market. Users of the application should be able to obtain the following information:

1. Future price changes of London properties that fall within a user-defined search radius, which can be fine-tuned by options such as property type
2. Locations that can be considered within a search radius, given a purchasing budget from the user

Obtaining predictions through the mobile application should be quick for the users, even though the prediction mechanism may involve computationally intensive tasks. We plan to achieve this through a client-side and server-side implementation for the application.

The client-side should be responsible for presenting the predictions through visualisations that highlight the geographical variations of our predictions. The client should be built upon a widely accessible platform so that our application can reach the largest audience. On the other hand, the server-side should be responsible for the prediction system, where training of the dataset takes

place in background, and the handling of prediction queries from the client. It should be built upon a server which has the necessary computing power and implementation libraries installed.

Besides being concerned with the implementation of the application, we would also need to derive an optimal prediction method for future London housing prices. Various machine learning techniques will be explored, tested with past data on London property transactions, before selecting one and modifying it to best suit our needs.

Chapter 2

Background

In this chapter, we first give an overview of the housing market in London. We will look at the dataset that we will be using in this project, and explain the features we have chosen to use in the prediction model. Lastly, we discuss related work that we have discovered during our initial research and its relevance to the project.

2.1 London Housing Market

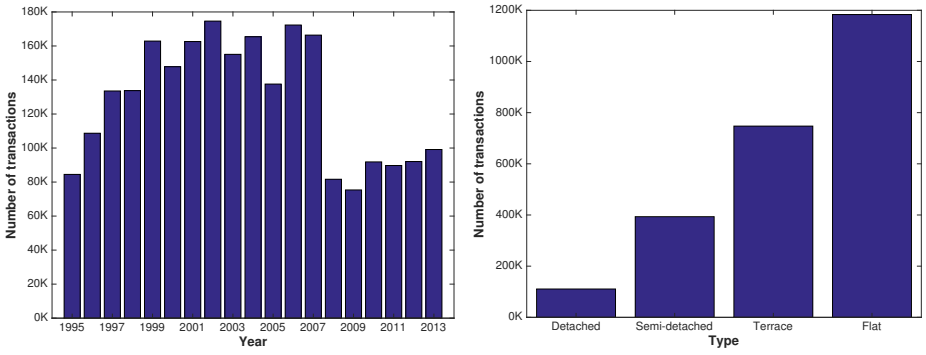
In the last decade, the housing market in London has been rapidly growing, with average housing prices increasing by more than 10% yearly on most years [3]. Together with stronger economic growth and increasing price expectations, this presents good news to current homeowners and potential homebuyers looking for a safe long-term investment.

However, London's housing market is not without volatility. The 2008/09 global financial crisis has seen a temporary decline of 8.4% in London housing prices in those years [4]. Even though political uncertainty before the May 2015 General Elections has led to a slower growth in Q1 2015, the market is picking up strongly in the recent months [5], with asking prices jumping up by 17% compared to before the elections [6].

Besides overall trends, the price of a house in London can vary greatly depending on the location. Properties in Inner London fetch much higher prices compared to properties in Outer London. The concentration of different types of houses also vary across London — houses in the central are mostly flats, while houses in the outskirts are mostly detached. Given the density of the properties in the capital, it is prime for us to model the relationship between housing prices and the various factors that affect it.

2.2 Dataset

Our initial dataset is obtained from the London Datastore[1], which contains details of all property transactions that have taken place in Greater London from 1995 to 2013. The total number of property transactions exceeds 2.4 million in the period. Further analysis of the dataset shows that the bulk of the property transactions happened during 1999 to 2007 (Figure 2.1a), and a majority of the transactions are flats (Figure 2.1b).



(a) Year of transaction (b) Property type

Figure 2.1: Number of property transactions sorted by year of transaction and property type.

For each property transaction, various information are provided and can be categorised as follows:

- ID (Transaction ID)
- Date (Date processed, Month of transaction, Year of transaction, etc)
- Transaction Price
- Property classification (Type, Build, Tenure)
- Address information (Postcode, Local authority, Full address, Borough, Ward, etc)

Out of all the information available to us, we have identified relevant features to be used in our prediction model, as we believe these variables contribute to the price of a house. Table 2.1 shows our choice of independent variables and dependent variable. We have chosen to represent an address by its postcode as it is a concise and unique reference to the location of each property. Further information on how we prepare the variables for use in our prediction model is explained in Section 3.1.

[1] http://data.london.gov.uk/dataset/average-house-prices-borough

Independent Variables	Dependent Variable
Date of transaction (Month, Year) Type (Detached/Semi-detached/Terrace/Flat) Build (Old/New) Tenure (Leasehold/Freehold) Address (Postcode)	Price of property (£)

Table 2.1: Variables to be used in our prediction model

As the London Datastore no longer maintains a repository of London-only property transactions after 2013, we obtain the dataset of property transactions after 2013 directly from the Land Registry. With their online search tool[2], we make a query for all property transactions in London starting from 2014. After which, we select only the relevant data and merge them with our initial dataset.

2.3 Related Work

After some research, we have found several related work that attempt to predict London housing prices, which we explore in this section. We consider what has been done and what can be learnt from them.

2.3.1 Property Agents and Economists

The job of predicting UK housing market usually falls within the domain of property agents such as Foxtons[3], Savills[4] and economic departments in leading universities. Predictions are made through a variety of factors: market sentiments, market data, economic performance, surveyor valuations, inflation, etc [7]. They often cover the whole of UK and forecast values are segregated by regions and major cities [8]. These predictions are often published in the news, or through publications made available to property investors. Figure 2.2 shows one such example of a publication on predictions of UK housing prices.

However, these predictions are often generic and averaged across different property types and locations within a certain region. Future prospects for a semi-detached house in a desired suburb are likely to differ from that of a flat in a less-desired suburb. Knowing the general forecast for London housing prices would not be able to help a homebuyer decide on which location in London or which property type he/she can consider. Moreover, the fact that these predictions are often made available through mass media does not allow for customisability and interactivity from the users.

[2]http://landregistry.data.gov.uk/app/ppd
[3]http://www.foxtons.co.uk/
[4]http://www.savills.co.uk/

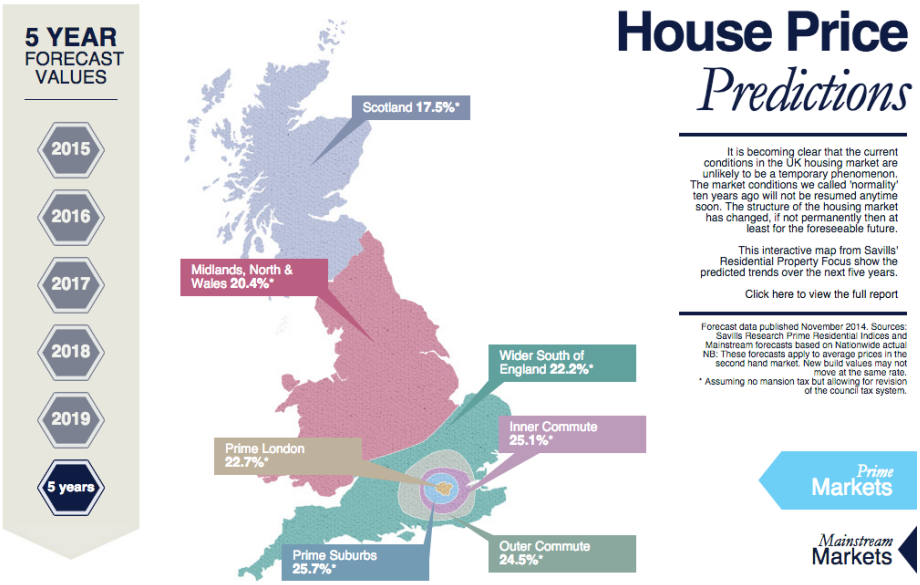


Figure 2.2: A publication on UK housing price predictions for the next 5 years by Savills PLC. [9]

2.3.2 Property Search Websites

Having millions of properties listed allow property search websites such as Zoopla[5] to come up with a prediction model for the current value of every property. Data is being collected from various sources such as estate agents, website users, surveyors and the government [10]. While the actual algorithm is proprietary, it is mentioned that the algorithm considers factors [10] such as:

1. Past, present and asking prices for the specific property and similar properties around it
2. Various characteristics such as size, type and number of rooms
3. Changes in market values of similar houses in the local area

However, it is to be noted that Zoopla estimates are specific to each property, not by districts or regions, as seen in Figure 2.3. Moreover, the estimates only refer to current values, which does not allow users to see the potential growth of property value in the future. As such, the calculations do not cater to users looking for long term forecasts of housing prices in a given area or of a certain type, which is the problem our project plans to tackle. Interestingly, Zoopla provides a confidence level for their estimates, which will be a useful benchmark to test our predictions against.

[5]http://www.zoopla.co.uk/

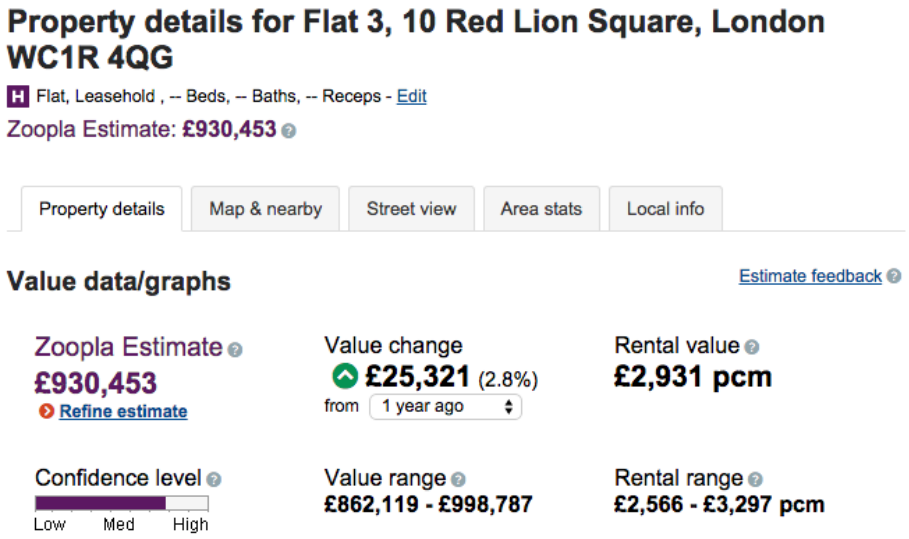


Figure 2.3: Example of a Zoopla estimate for a London property

2.3.3 Summary

One advantage that the property agents and property search websites have is the access to information that are not public. The public dataset from the Land Registry, as discussed in Section 2.2, does not include details such as the number of rooms or the size of the property, which we normally consider to be factors affecting house prices. As our prediction model relies solely on the public dataset, it may pale in comparison to these related work in terms of accuracy. However, none of them have reached what this project has set out to achieve: an application providing location-specifc predictions of housing prices within London that are calculated based on user-selected options.

Chapter 3

Regression Methods

In this chapter, we start by introducing the concept of machine learning and the role which regression plays. Then, we discuss why regression methods are what we need to predict future housing prices, before delving into details of various regression methods that have been explored during the course of this project. Each method contains fundamentals that build up to the next, making it easier to understand how we arrive at the method of our choice.

3.1 Introduction

Machine learning is the study of algorithms that can learn from data and make predictions, by building a model from example inputs rather than following static instructions [11]. These algorithms are typically classified into three categories: *supervised learning*, *unsupervised learning* and *reinforcement learning*.

In supervised learning, the system is presented with example inputs and outputs, with the aim of producing a function that maps the inputs to outputs. Regression and classification problems are the two main classes of supervised learning [12]. Unsupervised learning is concerned with leaving the system to find a structure based on the inputs, hopefully finding hidden patterns. Examples include density estimation [13, 14], dimensionality reduction [15, 16] and clustering [17]. Lastly, reinforcement learning is the study of how a system can learn and optimise its actions in an environment to maximise its rewards [18], such as training a robot to navigate a maze.

Regression is a subset of supervised learning, where the outputs are continuous. The problem of predicting future housing prices can be considered a regression problem, since we are concerned with predicting values that can fall within a continuous range of outputs. Through regression, we will be able to explore the relationship between the independent variables we have selected

(date, type, build, tenure, address) and the property price.

3.2 Linear Regression

The simplest regression model is *linear regression*, which involves using a linear combination of independent variables to estimate a continuous dependent variable [11]. While the model is too simple to accurately model the complexity of London housing market, there are many fundamental concepts in linear regression that many other regression techniques build upon.

If we consider a model where y is the dependent variable and $\mathbf{x} = (x_0, \ldots, x_D)^T$ is the vector of D independent variables, a linear regression model can be formulated as follows:

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) + \varepsilon_j, \tag{3.1}$$

where $\phi_j(\mathbf{x})$ is a *basis function* with corresponding parameter w_j, ε_j is the noise term, M is the number of basis functions and $\mathbf{w} = (w_0, \ldots, w_{M-1})^T$. Assume the noise terms ε_j are independent and identically distributed, where $\varepsilon_j \sim \mathcal{N}(0, \sigma^2)$. Basis functions can either be linear functions of the independent variables in the simplest case where $\phi_j(x) = x_j$, or extended to non-linear functions such as $\phi_j(x) = x_j^2, x_j^3$ or $\sqrt{x_j}$. In most cases, $\phi_o(x)$ is set as 1, such that w_0 can act as a bias parameter to allow for fixed offset in the data [19]. Even though $y(\mathbf{x}, \mathbf{w})$ can be a non-linear function of input $\mathbf{x}$ through the use of non-linear basis functions, the model is still linear with respect to $\mathbf{w}$. Alternatively, we can also express (3.1) as

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) + \boldsymbol{\varepsilon}, \tag{3.2}$$

where $\boldsymbol{\phi} = (\phi_0, \ldots, \phi_{M-1})^T$ and $\boldsymbol{\varepsilon} = (\varepsilon_0, \ldots, \varepsilon_{M-1})^T$, which simplifies the expression and allow us to code in vectorised form, thereby avoiding the need for loops in computation.

3.2.1 Error Function and Maximum Likelihood Solution

In order to fit a linear regression model to the dataset, we need to minimise an error function. A common error function of choice is the sum-of-squares error function, which takes the form

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2, \tag{3.3}$$

where $\mathbf{w}^T\phi(\mathbf{x}_n)$ is the predicted value, y_n is the actual value and N is the size of the data set. From a probabilistic perspective, we can also maximise a likelihood function under the assumption of a Gaussian noise model, and prove that it is equivalent to minimising the error function [20].

Firstly, we express the uncertainty associated with the actual value y_n using a Gaussian distribution with inverse variance (precision) β, in the form

$$p(y_n|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(y_n|\mathbf{w}^T\phi(\mathbf{x}_n), \beta^{-1}\mathbf{I}). \tag{3.4}$$

Considering inputs $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ and actual values $\mathbf{y} = \{y_1, ..., y_N\}$, the likelihood function can be constructed from this relationship

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(y_n|\mathbf{w}^T\phi(\mathbf{x}_n), \beta^{-1}\mathbf{I}). \tag{3.5}$$

To determine the optimal values for $\mathbf{w}$ and β, we can maximise the likelihood function with respect to $\mathbf{w}$. This can be done by taking the derivative of the logarithm of the likelihood function, which results in

$$\nabla_{\mathbf{w}} \log p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \beta) = \beta \sum_{n=1}^{N} \{y_n - \mathbf{w}^T\phi(\mathbf{x}_n)\}\phi(\mathbf{x}_n)^T. \tag{3.6}$$

If we take the derivative of the error function (3.3), we can see that it is equivalent to (3.6), thus proving that maximising the likelihood function is equivalent to minimising the error function.

Before solving for $\mathbf{w}$, let us define $\mathbf{\Phi}$, a $N \times M$ *design matrix*, whose elements are given by the basis functions applied to the matrix of values of the input variables, such that

$$\mathbf{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \dots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \dots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \dots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}. \tag{3.7}$$

By setting (3.6) to zero and solving for $\mathbf{w}$, we are able to obtain a closed-form solution, which takes the form

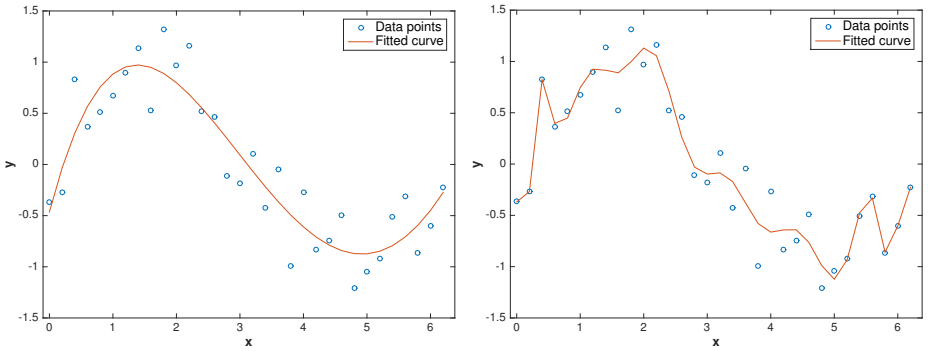
$$\mathbf{w_{ML}} = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{y}. \tag{3.8}$$

This is known as the *normal equation*. Solving for $\mathbf{w}$ through the normal equation will give us the weight vector that best fits the data.

3.2.2 Regularisation

Even though minimising the error function can solve the problem of finding $\mathbf{w}$, we have yet to solve the problem of choosing appropriate basis functions $\phi(\mathbf{x})$. Since basis functions are often in polynomials of the input variables, we can reduce the problem to finding the appropriate order M of the polynomial.

While choosing a higher order M introduces flexibility into the model and is likely to result in a better fit to the dataset, using a high order polynomial can sometimes result in *over-fitting*. Over-fitting occurs when the model fits the noise in the data instead of generalising [11].



(a) $M = 3$ (b) $M = 10$

Figure 3.1: Plots of polynomial models of different orders (red line) fitted to the dataset (blue points). Dataset is generated from a sinusoidal function.

This is illustrated in Figure 3.1, where it can be seen that choosing a polynomial degree of $M = 10$ results in a model that passes through more data point, but oscillates rapidly. This is likely to happen when the data used to train the model, known as the training set, is limited in size and an overly complex model is used. While over-fitted models may fit the training set very well (low training error), they are likely to have poor predictive performance on unseen data (high test error). A model that generalises well should have a low test error, which is what we are interested in.

To curb the problem of over-fitting, a regularisation term can be introduced to the error function. This takes the form of

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y_n - \mathbf{w}^T\phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}, \tag{3.9}$$

where λ is the *regularisation coefficient*. Adding a regularisation helps to prevent over-fitting by penalising large coefficient values. When applied to the closed-

form solution, the new normal equation can be defined as

$$\mathbf{w} = (\lambda \mathbf{I} + \mathbf{\Phi}^T \mathbf{\Phi})^{-1} \mathbf{\Phi}^T \mathbf{y}. \tag{3.10}$$

3.3 Bayesian Linear Regression

In our discussion of linear regression, we aim to determine $\mathbf{w}$, the weight vector which assigns a weight to each basis function that results in a best-fit of the data. However, the regression model will never be perfect, which implies a certain degree of error associated with every prediction made. By turning to a *Bayesian treatment* of linear regression [21], we will be able to express both the model uncertainty and measurement uncertainty. This will help us in constructing confidence intervals for our predictions of future property prices.

3.3.1 Posterior Distribution

We begin by defining a prior probability distribution over $\mathbf{w}$, which is given by a Gaussian distribution of the form

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0), \tag{3.11}$$

where $\mathbf{m_0}$ is the mean and $\mathbf{S_0}$ is the covariance. The posterior distribution is defined to be proportional to the product of the prior and the likelihood function [11], in the form

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{w}) p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})}, \tag{3.12}$$

where $p(\mathbf{y}|\mathbf{X})$ is the marginal likelihood. Using the likelihood function constructed in (3.4), the posterior distribution can be derived to form

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N), \tag{3.13}$$

where

$$\mathbf{m}_N = \mathbf{S}_N(\mathbf{S}_0^{-1}\mathbf{m}_0 + \beta \mathbf{\Phi}^T \mathbf{y}), \tag{3.14}$$

$$\mathbf{S}_N^{-1} = \mathbf{S}_0^{-1} + \beta \mathbf{\Phi}^T \mathbf{\Phi}. \tag{3.15}$$

A common choice for a prior, which we shall be using, is a zero-mean Gaussian with a precision parameter of α ($p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$), which results in a posterior distribution of

$$\mathbf{m}_N = \beta \mathbf{S}_N \mathbf{\Phi}^T \mathbf{y}, \tag{3.16}$$

$$\mathbf{S}_N^{-1} = \alpha\mathbf{I} + \beta\mathbf{\Phi}^T\mathbf{\Phi}. \tag{3.17}$$

3.3.2 Predictive Distribution

More importantly, we would like to determine the predictive distribution, the probability distribution when we predict t_* for new values $\mathbf{x}_*$. This is defined by the convolution of the conditional distribution of the observed values and the posterior distribution [11], resulting in

$$p(t_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y}, \alpha, \beta) = \int p(t_*|\mathbf{X}, \mathbf{w}, \beta)p(\mathbf{w}|\mathbf{X}, \mathbf{y}, \alpha, \beta)d\mathbf{w} \tag{3.18}$$

$$= \mathcal{N}(t_*|\mathbf{m}_N^T\phi(\mathbf{x}_*), \sigma_N^2(\mathbf{x}_*)) \tag{3.19}$$

where the variance takes the form

$$\sigma_N^2(\mathbf{x}_*) = \frac{1}{\beta} + \phi(\mathbf{x}_*)^T\mathbf{S}_N\phi(\mathbf{x}_*). \tag{3.20}$$

3.4 Relevance Vector Machines

Keeping in mind the advantages of having posterior probabilities, we can now explore the *relevance vector machine* (RVM) [22, 23]. The RVM for regression mirrors the linear regression model, with basis functions replaced by *kernels* [24]. A kernel function is defined as an inner product in the feature space

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T\phi(\mathbf{x}), \tag{3.21}$$

where $\phi(\mathbf{x})$ is a fixed feature space mapping. Through kernels, we can extend models to represent data in high-dimensional feature space, also known as the *kernel trick* [25]. While a kernel can be manually constructed by defining a $\phi(\mathbf{x})$ and taking the inner product, any similarity function on pairs of data points in the original space can also be a valid kernel. However, this similarity function has to satisfy *Mercer's condition*, which states that

$$\iint k(\mathbf{x}, \mathbf{x}')g(\mathbf{x})g(\mathbf{x}')d\mathbf{x}d\mathbf{x}' \geq 0, \tag{3.22}$$

for any $g(\mathbf{x})$ such that $\int g(\mathbf{x})^2 d\mathbf{x}$ is finite [26]. By using kernels, we can overcome the limited expressiveness of a Bayesian linear model, where the feature vector has to be explicitly transformed into higher dimensions. In most cases, this is also computationally cheaper than calculating the coordinates of data in the transformed feature space.

The RVM is based upon the widely successful approach of *support vector*

machines (SVM) [27, 28] in supervised learning, with the main advantage of according a predictive likelihood for every prediction. It also typically results in much sparser models than SVM, giving faster performance in the validation and testing phase while maintaining comparable generalisation performance [23].

A common choice for kernels is the squared exponential (SE) kernel, due to its flexibility to work in infinite dimensions. It takes the form

$$k(\mathbf{x}, \mathbf{x}') = \exp(-||\mathbf{x} - \mathbf{x}'||^2/2\sigma^2), \tag{3.23}$$

where σ is the kernel width. As a result, the linear model for RVM can now be defined in the following form

$$y(\mathbf{x}) = \sum_{j=1}^{M} w_j k(\mathbf{x}, \mathbf{x}_j) + \varepsilon_j. \tag{3.24}$$

The prior in RVM typically uses a separate hyperparameter α_i for each weight parameter ω_i. The prior takes the form

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_{i=1}^{M} \mathcal{N}(w_i|0, \alpha_i^{-1}), \tag{3.25}$$

where $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_M)^T$. Using the result (3.13) we obtained earlier in Bayesian linear regression, we can derive the posterior distribution for the weights, which is defined by

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}, \alpha, \beta) = \mathcal{N}(\mathbf{w}|\mathbf{m}, \boldsymbol{\Sigma}), \tag{3.26}$$

where

$$\mathbf{m} = \beta \boldsymbol{\Sigma} \boldsymbol{\Phi}^T \mathbf{y}, \tag{3.27}$$

$$\boldsymbol{\Sigma} = (\mathbf{A} + \beta \boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1}. \tag{3.28}$$

where $\mathbf{A} = diag(\alpha_i)$, β is the noise precision parameter and $\boldsymbol{\Phi} = \mathbf{K}$, a $N \times N$ kernel matrix with elements $k_{nm} = k(x_n, x_m)$.

3.4.1 Marginal Likelihood Maximisation

To determine $\boldsymbol{\alpha}$ and β, we can maximise the marginal likelihood, which results in an automatic trade-off between data fit and model complexity. The maximisation of the marginal likelihood is done by integrating out $\mathbf{w}$, and the logarithm

of it is in the form

$$\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\alpha}, \beta) = \log \int p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \beta) p(\mathbf{w}|\boldsymbol{\alpha}) d\mathbf{w} \quad (3.29)$$

$$= -\frac{1}{2}[N \log(2\pi) + \log |\mathbf{C}| + \mathbf{y}^T \mathbf{C}^{-1} \mathbf{y}], \quad (3.30)$$

where $\mathbf{C}$ is a matrix given by

$$\mathbf{C} = \beta^{-1}\mathbf{I} + \mathbf{\Phi}\mathbf{A}^{-1}\mathbf{\Phi}^T. \quad (3.31)$$

Finally, by setting the derivative of (3.30) to zero, we can obtain the re-estimation equations for the hyperparameters $\boldsymbol{\alpha}$ and β

$$\alpha_i^{new} = \frac{\gamma_i}{m_i^2}, \quad (3.32)$$

$$(\beta_{new})^{-1} = \frac{||\mathbf{t} - \mathbf{\Phi}\mathbf{m}||^2}{N - \Sigma_i \gamma_i}, \quad (3.33)$$

where $\gamma_i = 1 - \alpha_i \Sigma_{ii}$, a quantity that determines the extent to which the corresponding w_i is determined by the data.

Having the re-estimation equations defined, the maximisation of the marginal likelihood can take place through a loop of calculating the mean and covariance of the posterior in (3.27) and (3.28), re-estimating $\boldsymbol{\alpha}$ and β in (3.32) and (3.33), and re-estimating the mean and covariance of the posterior again, until a chosen convergence criterion is reached.

However, this original approach suffers from $\mathcal{O}(N^3)$ computations in the first few iterations, as all N basis functions are considered initially in the model. An accelerated learning algorithm has been proposed based on this approach, where an "empty" model is initialised and basis functions are subsequently added [29].

Despite having a vector of hyperparameter α_i, the optimisation will result in a large proportion of them going towards infinity, which forces the corresponding weight parameters to have posterior distributions concentrated at zero. This will result in a sparse model mentioned earlier, as the corresponding basis functions play no role in the predictions. The remaining non-zero α_i, and their corresponding weights and basis functions, are thus called the *relevance vectors*.

3.4.2 Predictive Distribution

With the optimal hyperparameters $\boldsymbol{\alpha}^*$ and β^* determined, the predictive distribution is similar to that of Bayesian linear regression, given by

$$p(t_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y}, \boldsymbol{\alpha}^*, \beta^*) = \int p(t_*|\mathbf{X}, \mathbf{w}, \beta^*) p(\mathbf{w}|\mathbf{X}, \mathbf{y}, \boldsymbol{\alpha}^*, \beta^*) d\mathbf{w} \quad (3.34)$$

$$= \mathcal{N}(t_*|\mathbf{m}^T\phi(\mathbf{x}_*), \sigma^2(\mathbf{x}_*)), \tag{3.35}$$

where the variance takes the form

$$\sigma_N^2(\mathbf{x}_*) = \frac{1}{\beta^*} + \phi(\mathbf{x}_*)^T\mathbf{\Sigma}\phi(\mathbf{x}_*). \tag{3.36}$$

3.4.3 Problems with Predictive Variances

Despite being able to produce predictive variances, it has been shown that the predictive variances become smaller the further the test inputs are away from the relevance vectors [30, 31]. This means that predictions become more certain as we move away from the training data, which is counter-intuitive and undesirable. Such a problem occurs during the usage of *localised basis functions*, where basis functions are centered on each of the training data points. An example is the SE

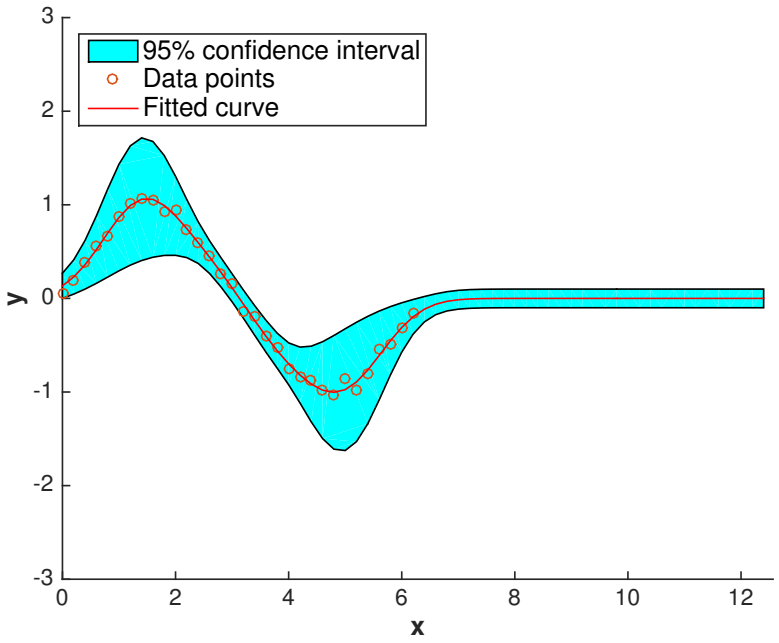


Figure 3.2: Predictive variance of RVM tends towards the noise level of the data as test inputs deviate further from the dataset.

kernel mentioned earlier in the section. When a test input lies far away from the relevance vectors, the basis functions each produce a small response to the input, resulting in a predictive mean of zero and variance close to the noise level [30]. This can be seen in Figure 3.2, where the predictive uncertainty, represented by the blue area, decreases to the noise level when predictions extend beyond the range of the training data. This would be of a major consideration in our

project, as we are concerned with predicting future housing prices that extend outside the range of our dataset.

However, this would not be an issue when we consider Gaussian processes, a model which uses infinitely many basis functions placed everywhere, not just at the training points, which we explore in the section below.

3.5 Gaussian Process

Unlike all the regression models discussed earlier which define a distribution over parameters, a Gaussian Process (GP) [32, 33] model defines a distribution over functions. A GP can be defined as a collection of random variables, where any finite number of which have a joint Gaussian distribution. Notationally, it takes the form

$$f(\mathbf{x}) \sim GP(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')), \tag{3.37}$$

where the mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$ are

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})], \tag{3.38}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]. \tag{3.39}$$

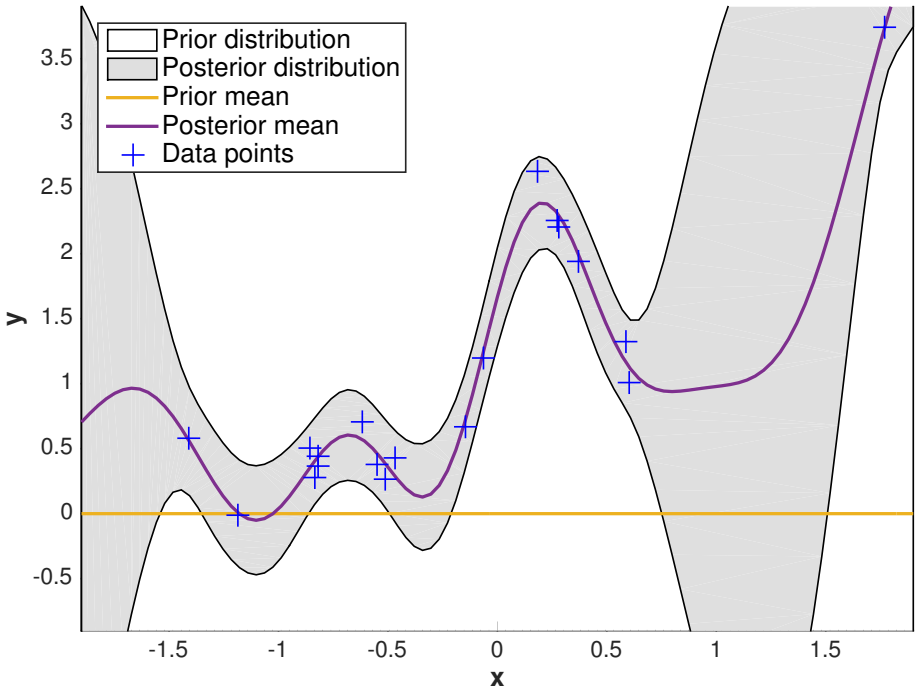


Figure 3.3: Posterior distribution and mean of a GP after the addition of observed data points.

In a GP, a prior distribution represents our initial beliefs on the type of mean function we expect to observe. Given a dataset of observations, we compute the posterior mean through the gradual addition of data points, which constrains the shape of the posterior mean. The combination of the prior distribution and the dataset leads to the posterior distribution, represented by the grey area in Figure 3.3. Notice how the posterior uncertainty is reduced close to the observed data points but falls back to the prior at areas without data points, which is a desirable situation that classical RVMs are not able to achieve.

In GPs, the covariance function, or kernel, encodes useful information about the function we are trying to model in a GP. For example, we can consider the SE kernel, as it is appropriate for modelling smooth functions. In the context of a GP with noisy observations, it takes the form

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 exp(-||\mathbf{x} - \mathbf{x}'||^2/2l^2). \tag{3.40}$$

The lengthscale l defines the smoothness of the function and the distance to which we can extrapolate from the training data. The larger the lengthscale value, the slower the function changes, resulting in a smoother function. Signal variance σ_f^2 defines the level of variation of the function from its mean. The larger the signal variance, the greater the amplitude. Together, these form the hyperparameters of the SE kernel, which define the properties of the function being modelled.

3.5.1 Training

Similar to RVM, we can determine the optimal hyperparameters of the SE kernel through the maximisation of marginal likelihood. With the likelihood function defined as $p(\mathbf{y}|\mathbf{X}, \mathbf{f}) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma_n^2\mathbf{I})$, where σ_n^2 is the noise variance, and using a zero-mean Gaussian prior $p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$, the marginal likelihood takes the form

$$p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{X}, \mathbf{f}) p(\mathbf{f}|\mathbf{X}) d\mathbf{f} = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K} + \sigma_n^2\mathbf{I}). \tag{3.41}$$

For numerical stability in optimisation, the logarithm of the marginal likelihood is normally taken instead, which is given by

$$\log p(\mathbf{y}|\mathbf{X}) = -\frac{1}{2}\mathbf{y}^\top(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K} + \sigma_n^2\mathbf{I}| - \frac{n}{2}\log 2\pi. \tag{3.42}$$

By seeking the partial derivatives of the log marginal likelihood with respect to the hyperparameters and equating it to zero, the hyperparameters can be set. This process can be facilitated through the use of gradient-based optimisers. A popular choice for use with GPs is the conjugate gradients method [34], a

prominent iterative method used to solve systems of linear equations which cannot be handled by other direct methods due to its large size.

3.5.2 Predictive Distribution

The joint distribution of the predictions from the training points, $\mathbf{f}$, and predictions from the new input points, $\mathbf{f}_*$ can be rewritten from (3.41) to form

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) + \sigma_n^2\mathbf{I} & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right), \tag{3.43}$$

where X is the set of training points, X_* is the set of test points, and $K(X, X_*)$ denotes the matrix of covariances evaluated at all pairs of points in X and X_*, similarly for the other entries in the joint distribution. The predictive distributions can be derived by obtaining a conditional distribution for $\mathbf{f}_*$ given $\mathbf{y}$, which gives

$$\mathbf{f}_*|X, X_*, \mathbf{y} \sim \mathcal{N}(\mathbf{f}'_*, cov(\mathbf{f}'_*)), \tag{3.44}$$

where

$$\mathbf{f}'_* = K(X_*,X)[K(X,X) + \sigma_n^2\mathbf{I}]^{-1}\mathbf{y}, \tag{3.45}$$

$$cov(\mathbf{f}'_*) = K(X_*,X_*) - K(X_*,X)[K(X,X) + \sigma_n^2\mathbf{I}]^{-1}K(X,X_*). \tag{3.46}$$

3.5.3 Distributed GPs

One fundamental weakness of GPs is that they scale badly with the size of the dataset. The training complexity is dominated by the inversion of $\mathbf{K} + \sigma_n^2\mathbf{I}$ in the marginal likelihood, which takes $\mathcal{O}(n^3)$ operations, while the predictive complexity is dominated by the computation of the predictive variance, which takes $\mathcal{O}(n^2)$ operations [35]. As a result, GPs have a practical limit of 10^4 for the dataset size [36]. This presents a problem if we were to use a GP to model our dataset on London housing prices, which has a size of about 2.4 million.

Various sparse approximations [37, 38] exist, whereby a subset of the data is used instead of the entire dataset. However, this raises the issue of how the subset should be selected, and the computations involved. Even with sparse approximations, the practical limit of the dataset size is limited to 10^6 [36].

With distributed GPs, we can handle dataset sizes of 10^7, by distributing computations to independent local "expert" models, where each model operates on a subset of the data [39]. Each expert, therefore, has its own set of hyperparameters to optimise. On top of handling larger dataset sizes, we plan to use distributed GPs in our project to exploit the spatial structure of our dataset by partitioning our data by location and training them independently. Sub-

sequently, predictions are made by combining the predictions from all experts [40, 41], taking into account the weight assigned to each prediction. However, this brings forth the issue of how the weights are to be assigned to each expert, which we shall explore below.

3.5.3.1 Product of Experts

The Product of Experts (PoE) model [39] proposes that the combined predictive distribution y_* at a test input x_* is computed as the product of all the predictive distributions of the experts, which takes the form

$$p(y_*|x_*, \mathcal{D}) = \prod_k p(y_*|x_*, \mathcal{D}^{(k)}) \tag{3.47}$$

$$\propto \mathcal{N}(y_*|\mu_*, \sigma_*^2), \tag{3.48}$$

where $\mathcal{D}^{(k)}$ is the k-th subset of the data. The product of all the Gaussian predictive distributions is also a Gaussian, where the mean and variance are

$$\sigma_*^2 = \left(\sum_k \sigma_k^{-2}(x_*)\right)^{-1}, \tag{3.49}$$

$$\mu_* = \sigma_*^2 \sum_k \frac{\mu_k(x_*)}{\sigma_k^2(x_*)}. \tag{3.50}$$

While the PoE model is straightforward to compute, it tends to produce overconfident models. As the number of experts increases, the combined prediction becomes more certain as the precisions add up, regardless of how certain each expert is (refer to (3.49)). Thus, using only the predictive variance of an expert is not necessarily the right measure for assigning weights to each expert [42].

3.5.3.2 Generalised Product of Experts

Improving upon the PoE model, the generalised Product of Experts (gPoE) model [42] proposes to weigh the responsibility of each expert by raising the predictive distribution to a power β_k, where the combined prediction distribution is defined as

$$p(y_*|x_*, X, \mathcal{D}) = \prod_k p_k^{\beta_k}(y_*|x_*, \mathcal{D}^{(k)}) \tag{3.51}$$

$$\propto \mathcal{N}(y_*|\mu_*, \sigma_*^2). \tag{3.52}$$

This gives the model the flexibility to increase or decrease the responsibility of each expert. As with the PoE model, the resulting predictive distribution is still

a Gaussian, with a mean and variance

$$\sigma_*^2 = \left(\sum_k \beta_k \sigma_k^{-2}(x_*) \right)^{-1} \tag{3.53}$$

$$\mu_* = \sigma_*^2 \sum_k \frac{\beta_k \mu_k(x_*)}{\sigma_k^2(x_*)}. \tag{3.54}$$

It can be seen that β_k serves as a scaling factor for the predictive variance. However, one has to ensure $\sum_k \beta_k = 1$ for the model to fall back to the prior.

3.6 Summary

Looking back at the regression models discussed, we have seen how a GP has overcome the problems associated with other models, which will be of concern to the prediction of future housing prices if not resolved. By using a GP as our choice of prediction method, we will be able to produce predictive distributions instead of just point estimates, provide reasonable predictive uncertainty when predictions are extrapolated to future dates, and harness the power of kernels to express features in higher dimensions. The specific implementation of GP to our project will be further explored in Chapter 4.

Chapter 4

Design and Implementation: Prediction System

In this chapter, we discuss the design and implementation of our prediction system for London housing prices, based upon a Gaussian process model. This includes how the dataset is being prepared, the implementation language and library used for training the dataset, and how predictions are made to suit the needs of the project.

4.1 Data Preparation

To prepare the dataset for the prediction system, some changes were made:

1. To capture geographical variation in housing prices more accurately, we decide to represent the address of the property in terms of latitudes and longitudes rather than postcodes. This is preferable to postcodes as neighbouring regions do not necessarily share similar postcodes (e.g. SW7 and W8). This process is automated through a Python script which looks up the postcode in a pre-compiled CSV file of London postcodes[1], and returns the corresponding latitude and longitude.

2. The date of transaction is now represented as the number of months since January 1995 (the earliest month of transaction in our dataset), to simplify the representation of time by months only, instead of month and year.

3. Binary categorical variables (build & tenure) are represented using one binary digit (i.e. (Build) 0 = Old, 1 = New / (Tenure) 0 = Leasehold, 1 = Freehold), while property type is represented using three binary digits

[1] http://www.doogal.co.uk/PostcodeDownloads.php

through the use of dummy coding [43] (i.e. 000 = Detached, 100 = Semi-detached, 010 = Terrace, 001 = Flat).

4. As the price of properties are often quoted in thousands, we have rounded our dependent variable to the nearest thousand, which also helps with the numerical stability of the model.

The prepared dataset and the corresponding values that each variable contains is shown in Table 4.1.

Independent Variables	**Dependent Variable**
Date of transaction (Months since 1995)	Price of property (£ '000s)
Type (000/100/010/001)	
Build (0/1)	
Tenure (0/1)	
Address (Latitude, Longitude)	

Table 4.1: Variables to be used in our prediction model

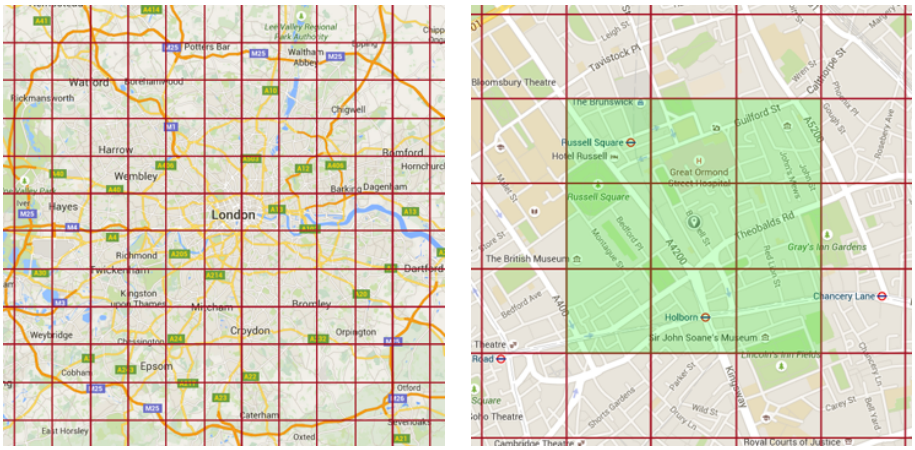
4.1.1 Utilisation of Data

Due to the huge number of data points (2.4 million), training a GP model using the entire dataset would be computationally intractable. Yet, limiting the size of our dataset to the practical upper bound of 10^4 would mean using less than 1% of the entire dataset, which is undesirable. As such, we will be using the concept of distributed GPs in our prediction model, to maximise the utilisation of our dataset and provide a way to combine local models. The concept of using local models to exploit the spatial properties of our dataset makes sense too.

To create local models, we have partitioned the dataset into multiple subsets by location. This is done by drawing a square grid on the map of London (Figure 4.1a), and grouping data points in a single box into a subset. Training will be thus be done on each subset independently.

Despite partitioning our dataset, we are still faced with the problem of having a disproportional number of data points across different subsets. Areas in central London can have datasets with more than 10,000 data points, while areas out of central London can have datasets below 100 data points. Regardless of the granularity of the square grid, this means that we have to maintain a practical upper and lower bound to the number of data points in each subset.

Taking into account the number of data points eventually used, the granularity of the grid and time taken for training, we propose to partition the map of London into a 50 x 100 square grid of length 1km each, where subsets have a lower and upper bound of 200 and 2,000 data points respectively. Subsets with data points below the lower bound will not be considered in our model,



(a) (b)

Figure 4.1: (a) Illustration of a square grid over the map of London. (b) Predictions made from points in the centre grid will only consider the direct neighbours and itself, which is highlighted in green.

whereas a random sample of 2,000 data points will be taken for each subset with data points above the upper bound. This brings us to a total of **2,057 usable subsets** and a **dataset utilisation of 90.1%**. For each subset, we have also taken 10% of the data as our validation set, while the remaining 90% serves as the training set.

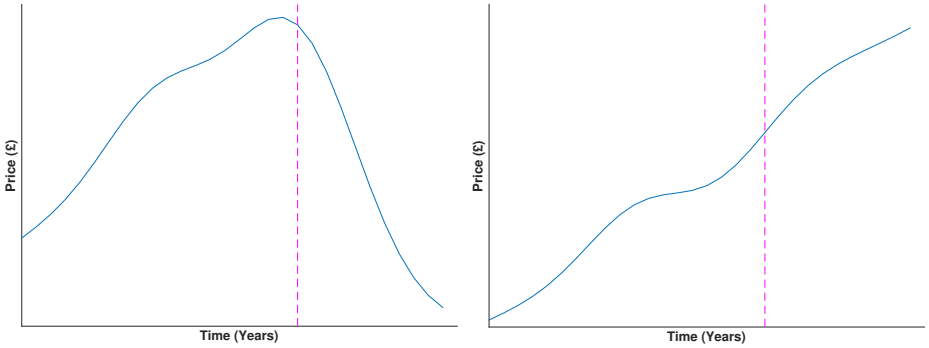
4.2 Training

To implement Gaussian process, we have chosen the GPML toolbox[2] which runs on MATLAB, due to its simple yet extendable format and a large library of covariance and mean functions. MATLAB is our preferred implementation language due to its performance and extensive support in matrix algebra, which is particularly relevant in a machine learning project.

4.2.1 Linear Prior Mean Function

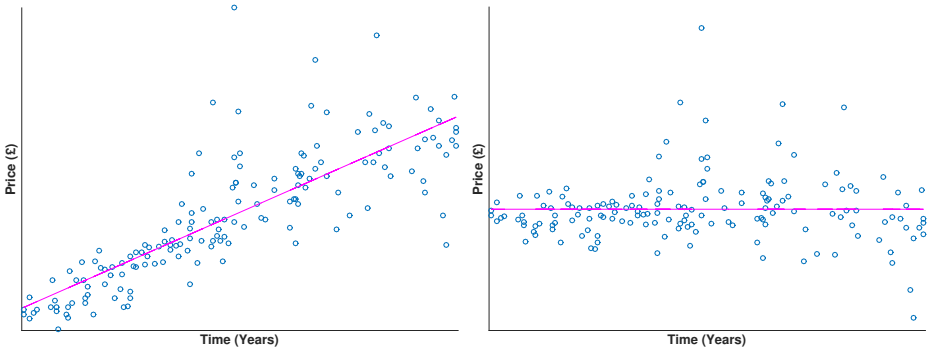
When test inputs of a GP lie outside the range of data used in training, the predictive means would tend towards the prior. With a fixed zero-mean Gaussian prior, predictive means of such test inputs would therefore tend towards zero. This leads to an undesirable behaviour where predictions of future housing prices tend towards zero with time, as seen in Figure 4.2a.

[2]http://www.gaussianprocess.org/gpml/code/matlab/doc/index.html



(a) Without linear prior mean (b) With linear prior mean

Figure 4.2: Housing prices predicted by the model using a zero-mean Gaussian prior and linear prior respectively. The magenta line is the date up to which the data is trained upon.

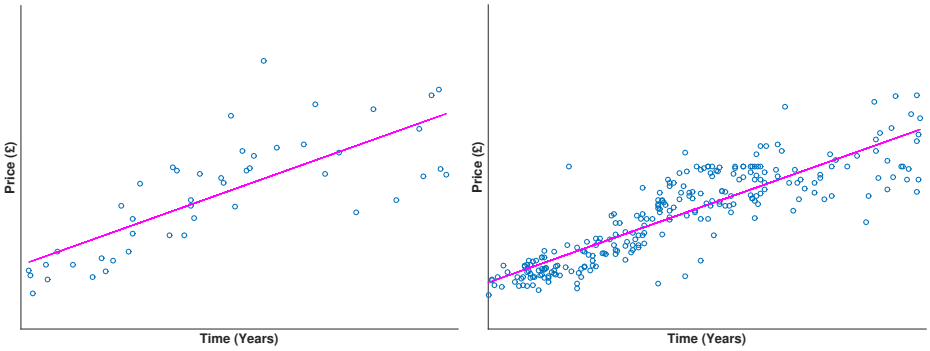


(a) Before adjustment (b) After adjustment

Figure 4.3: Training data from a sample area plotted against the date of purchase before and after adjustment using a linear prior mean. The magenta line represents the prior mean function with respect to time.

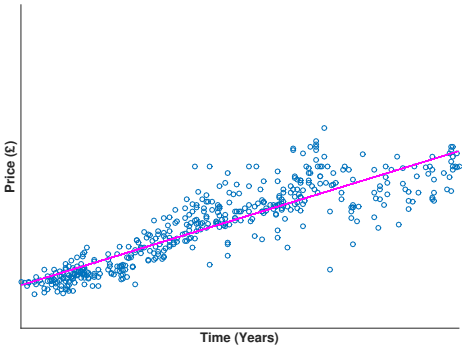
As such, we propose using a linear prior mean function that models the relationship between the date of purchase and price paid in our model (Figure 4.3a). The training data is adjusted by the linear prior mean function before the model is being trained (Figure 4.3b). In other words, we are modelling the variation of house prices along the straight line instead of modelling the trend in housing prices directly. Predictions of future housing prices will therefore tend towards the straight line fit gradually with time, instead of falling to zero (Figure 4.2b). In this approximation, we assume a (positive) linear relationship between date of purchase and price paid.

To finetune our approximation, we use a separate linear prior mean function for each house type, since the average price and trend can be very different

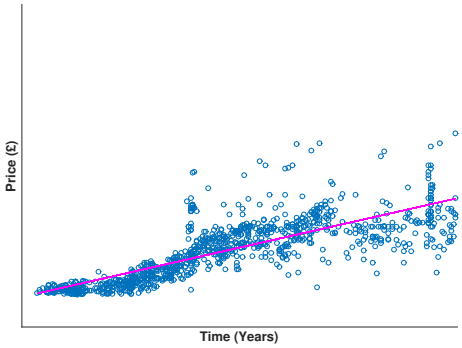


(a) Detached

(b) Semi-detached



(c) Terrace



(d) Flat

Figure 4.4: Plots of training data from a sample area sorted by property type. The magenta line represents the best linear fit for the data with respect to time, which varies across different property types.

across different house types. Figure 4.4 shows how the optimal linear fits can vary across different house types. The gradient and intercept for the prior mean of each model will be stored and used to adjust the predictions during the prediction phase.

4.2.2 Kernel

In our model, we have chosen to use a SE kernel, which we have defined previously in (3.40), due to its flexibility and ability to model smooth functions. Even though the optimal hyperparameters will be determined through a gradient-based optimiser (using conjugate gradients method) that is in-built in the GPML toolbox, we have initialised the hyperparameters to suitable values in (4.1) to aid the optimiser in reaching the ideal optima. The data has been normalised before arriving at these initial values.

$$l = \frac{1}{4}, \sigma_f^2 = \text{Var}[\mathbf{y}], \sigma_n^2 = \sigma_f^2/100. \tag{4.1}$$

4.3 Prediction

Once all the local models have been trained independently, predictions for each test input are made by combining the predictions of all local models. However, this can be an expensive operation, considering that we have over 2,000 areas to consider. Instead of combining the predictions of all areas, we propose to only combine the predictions from areas that are direct neighbours, in terms of location, to the area where the test input lies in (Figure 4.1b). Figure 4.5 illustrates the difference between the training and prediction phase of our model — we only consider a subset of all the local models during prediction. This exploits the spatial structure of the data and the corresponding local models we have defined. This solution serves as a compromise between speed and accuracy, with the assumption that the housing prices in an area is largely affected by the areas surrounding it only (e.g. price of a house in Kensington is unlikely to be affected by the housing prices in Dagenham).

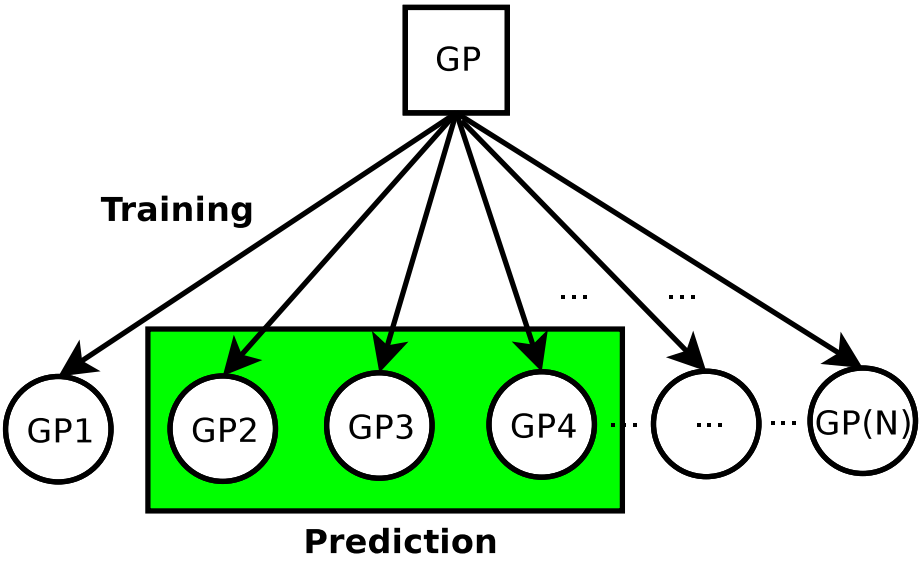


Figure 4.5: Distribution of GP to multiple independent local models during training. However, prediction is made from a subset of the local models.

To weigh the responsibility of each local model under the gPoE method, we propose to set β_k to be inversely proportional to the predictive variance (i.e. an expert with a lower predictive variance will have higher weightage). Keeping in

mind that $\sum \beta_k = 1$ in the gPoE model, we compute β_k by the equation

$$\beta_k = \frac{\sigma_k^{-2}}{\sum_{i=1}^{N} \sigma_i^{-2}}, \tag{4.2}$$

where σ_k^2 is the predictive variance and N is the number of experts considered. N would be 9 in our model, as we are only considering the direct neighbours of an area in a square grid.

Chapter 5

Design and Implementation: Mobile Application

In this chapter, we discuss the design and implementation of our prediction system of our mobile application. Our mobile application consists of both a client-side application with which the user interacts, and a server-side component where the prediction system resides and handles all queries from the client-side. We showcase the application's front-end design with screenshots, and also describe the system architecture on the server-side.

5.1 Client-side

We have decided to develop our mobile application using the Android platform[1], the leading mobile operating system in the world with a 78.0% market share, as of Q1 2015 [44]. As an open-source platform that is highly customisable, Android is currently used by top global smartphone makers, such as Samsung, LG and HTC in their flagship devices. Coupled with the fact that developers can easily distribute their applications through Google Play[2] at no cost, Android is the ideal platform for developing and distributing our mobile application to a large audience with minimal development effort.

5.1.1 Application Structure

Keeping in mind the objectives of our project, our mobile application has been designed with an application structure illustrated in Figure 5.1. From the home

[1] https://www.android.com/
[2] Google Play is Google's official application store and media distribution platform.

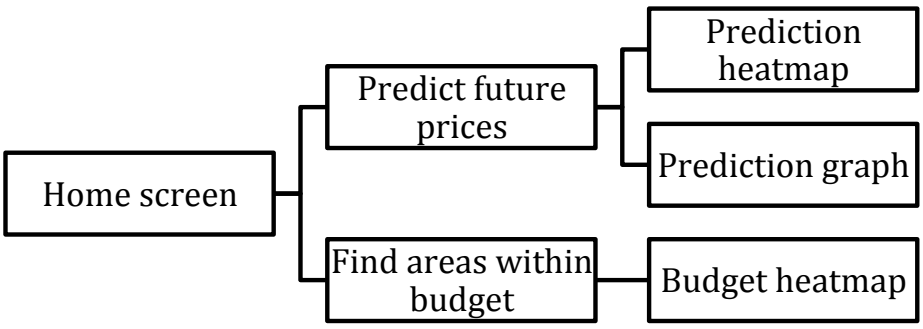


Figure 5.1: Application structure of our mobile application

screen, the user can either navigate to the "Predict future prices" module, or the "Find areas within budget" module.

The "Predict future prices" module allows the user to view future price predictions of London properties, given a search radius. The module will generate two heatmaps — one that highlights the predicted changes in housing prices, and one that highlights the actual predicted prices. However, since it would be difficult to display information about the confidence intervals of our predictions in a heatmap, the module will also generate a graph that highlights the confidence intervals and trend in housing prices in the specified area.

On the other hand, the "Find areas within budget" module allows the user to find out areas in London where they can consider purchasing a property in, given a budget. The module will subsequently generate a heatmap that highlights the suitable areas for consideration.

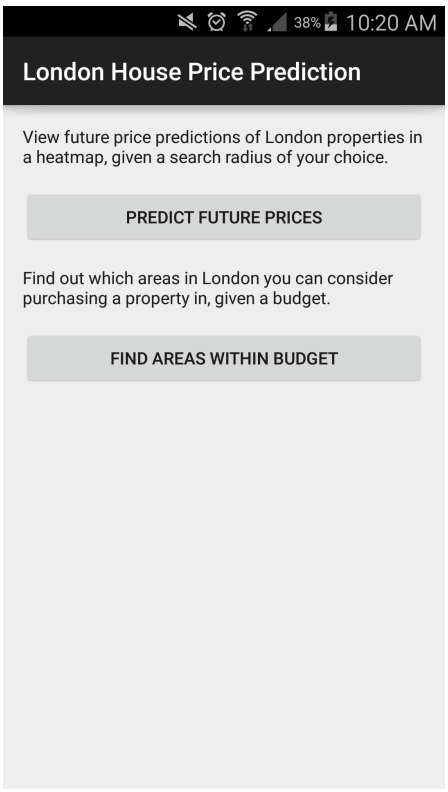
To support heatmap visualisations in our mobile application, we utilise the Google Maps Android API[3] which allow developers to use Google Maps as a mapping tool within an Android application with in-built visualisation libraries. By specifying a collection of latitudes and longitudes and their respective weights, the API can produce a heatmap visualisation on top of a Google Map embedded in an Android application, which makes it the ideal tool to generate the heatmaps.

Instead of using a client-side tool to generate the prediction graphs, we opted for a server-side implementation as there is already support for graphing within

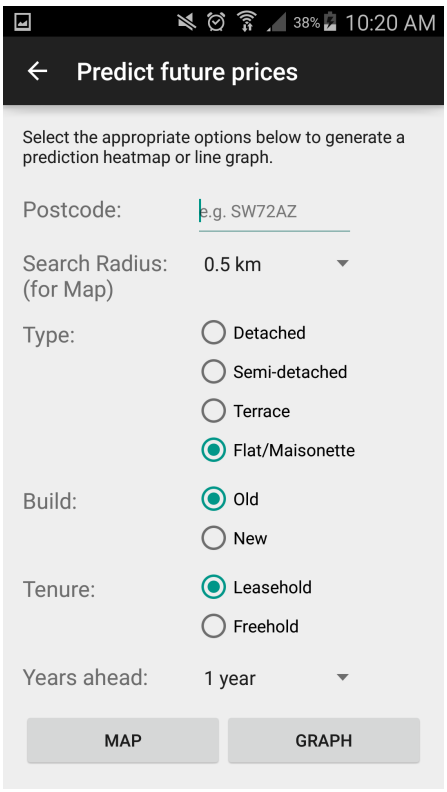
[3]https://developers.google.com/maps/documentation/android/start

MATLAB, which we are using for our prediction system in the server. Our mobile application will retrieve the graphs directly from the server, instead of retrieving textual data.

5.1.2 Application Design



(a) Home screen

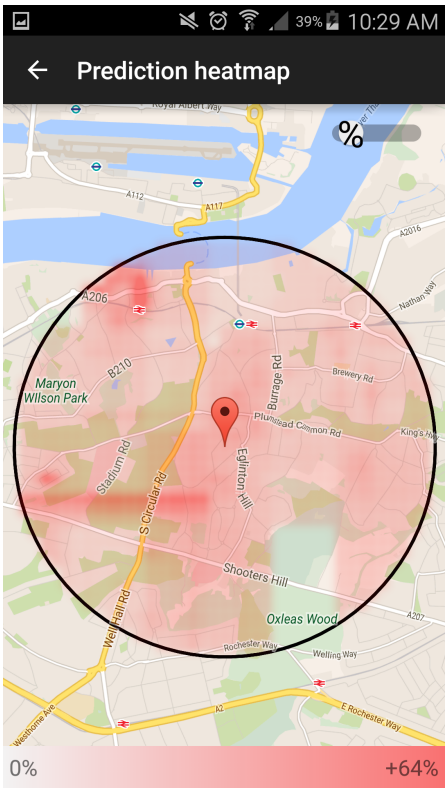


(b) "Predict future prices" module

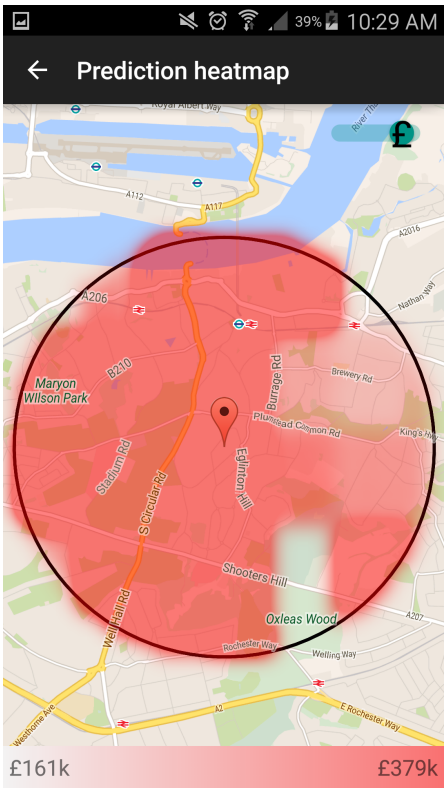
Figure 5.2: Screenshots of the home screen and "Predict future prices" module.

When a user opens the mobile application, he/she will be presented with a home screen (Figure 5.2a) with buttons to access both modules. A short description of each module is provided above each button to inform the user of the purpose of the module.

In the "Predict future prices" module (Figure 5.2b), the user will be asked to enter the parameters required to generate a prediction heatmap or graph. This includes the variables we use in our GP-based prediction method, the number of years in the future the prediction should consider and the search radius of the query (to be used for generating the heatmap). Once the user enters the



(a) Prediction heatmap (percentage change)

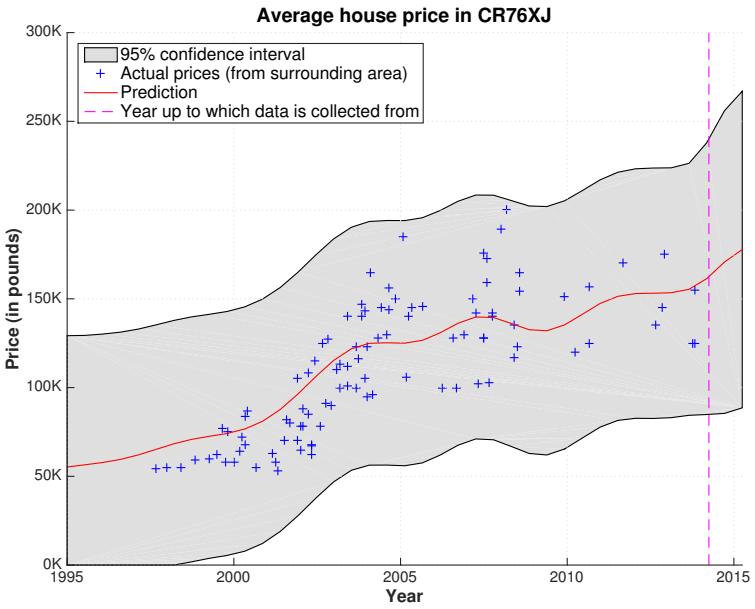


(b) Prediction heatmap (actual prices)

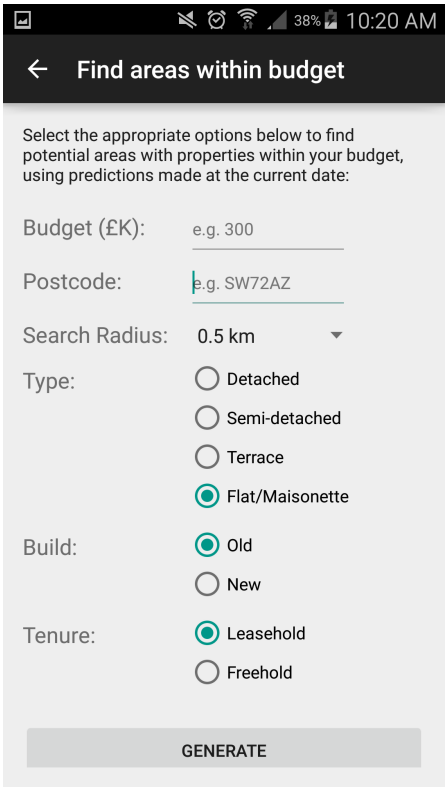
Figure 5.3: Screenshots of both prediction heatmaps.

parameters and click on the "Map" or "Graph" button, the application will check if the postcode entered is a valid London postcode. If valid, he/she will be brought to the prediction heatmap or graph respectively, otherwise an error message will appear requesting the user to enter a valid postcode.

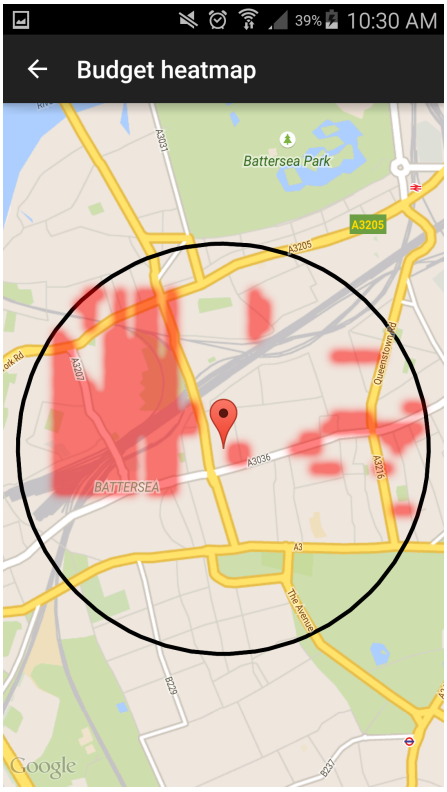
Once generated, the prediction heatmap (Figure 5.3a) helps the user to visualise the geographical variations in the changes in housing prices within the search radius, in terms of percentage increase and decrease. A percentage increase and decrease is represented in red and blue respectively. The intensity of the colour is proportional to the magnitude of change. To aid the user in understanding the heatmap, a colour scale is provided at the bottom of the screen. The user will be able to alter the zoom level of the heatmap through the "pinch" gesture, even though a minimum and maximum zoom level is applied to ensure the integrity of the colour scale of the heatmap. Users can also view a heatmap of actual predicted prices instead of percentage changes in prices (Figure 5.3b), through the switch button on the top right of the screen.



(a) Prediction graph



(b) "Find areas within budget" module



(c) Budget heatmap

Figure 5.4: Screenshots of prediction graph, "Find areas within budget" module and budget heatmap.

The prediction graph (Figure 5.4a) complements the heatmap, by displaying the trend in the average house price over the years, in the postcode provided by the user. The 95% confidence interval of our predictions is also included in the graph, to give the user an idea of how accurate our predictions are. Likewise, the user will be able to zoom in on the graph through the "pinch" gesture, as the details on the graph may not be visible to the user on mobile phones with limited screen space.

For the "Find areas within budget" module (Figure 5.4b), the same design principle applies as the "Predict future prices" module, albeit with an additional parameter for the user to enter their budget. In the budget heatmap (Figure 5.4c), areas highlighted in red represent areas which property prices are potentially below or equivalent to the budget instead.

5.2 Server-side

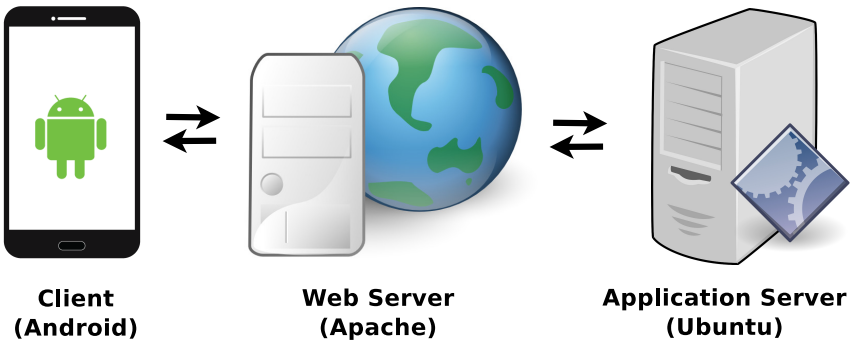


Figure 5.5: System architecture for our mobile application

The server-side of our mobile application consists of two components — a web server and an application server. The web server acts as the intermediary between the client and the application server, by handling HTTP requests from the client-side and returning generated data from the application server to the client. The application server is where the computations are handled, which includes our dataset, the prediction system described in Chapter 3, and other scripts to support it. Figure 5.5 shows a diagram of our system architecture.

5.2.1 Web Server

Our web server currently runs on a virtual machine hosted on DoC Private Cloud, a departmental Infrastructure-as-a-service (IAAS) platform, which allows us to easily create a virtual server that suits our needs. Ubuntu 14.04 LTS has been installed as the operating system and Apache HTTP Server software

has been installed to give the virtual machine the capabilities of a web server. Both Ubuntu and Apache have been chosen due to its widespread adoption and our prior experience using it.

To handle HTTP requests from the client and return dynamically generated content, we run Common Gateway Interface (CGI) scripts written in Bash on the web server. CGI scripts are sufficient for our needs as we are merely using the web server to pass on the parameters to the application server and return text content to the client.

```
http://XXX.XXX:XXX/predictmap.cgi?postcode=SW72AZ&radius
    =1&housetype=3&build=0&tenure=0&years=1
```

Listing 5.1: Sample HTTP request from client

A sample HTTP request from the client is shown in Listing 5.1, which is requesting to generate a prediction heatmap. The parameters and their corresponding values are passed into the CGI script, which are subsequently passed on to the application server by a call to the target MATLAB script to generate prediction values.

Communication with the application server has to be done through the Secure Shell protocol, due to our choice of using the departmental batch servers as our application server, which is explained in Section 4.2.2. Once the predictions are made in the application server, the text results are returned to the client as the response to the HTTP request.

5.2.2 Application Server

We are currently using one of the departmental batch servers to act as our application server. Operating on Ubuntu, the departmental batch servers have existing MATLAB licenses and come with hardware capable of running long programs spanning several hours or days. This makes it ideal for training our dataset, which takes over a day to train as there are over 2000 subsets, where each takes an average of 70 seconds to train.

New data is fed into the prediction system automatically on a monthly basis, through a system daemon that executes a script to retrieve new data on recently sold London properties. Data is obtained using a SPARQL query that calls the Land Registry console[4], with relevant variables selected and corresponding values prepared, before merging them into our dataset. Subsequently, the prediction system trains on the updated set of data, thereby ensuring our predictions are constantly refined with the latest data.

[4]http://landregistry.data.gov.uk/app/hpi/qonsole

Besides having the prediction system running on the prediction server, several Python scripts also exist on the application server to support the prediction system. These include helper scripts to convert a postcode into its corresponding latitude and longitude for input into the prediction system, as discussed in Section 4.1, and to populate a list of latitudes & longitudes during the generation of heatmaps. These scripts are implemented in Python instead of MATLAB as they do not require any form of matrix manipulation and Python will be able to execute these scripts more efficiently.

Chapter 6

Evaluation

We have managed to create a London house price prediction mobile application that has met the initial objectives of this project. However, the final product alone does not measure the performance and functionality of our project. In this chapter, we will be evaluating our project by looking into the quality of our prediction method, the performance of the prediction system and the functionality of the mobile application.

6.1 Prediction Method

To measure the quality of our GP-based prediction method, we use the *root mean squared error* (RMSE) and the *mean predictive log likelihood* (MPLL). RMSE is defined by the square root of the average of the squared residual $(y_* - f(\mathbf{x}_*))^2$ between the prediction value and actual value of each test point in our validation set. MPLL is defined by the average of the log predictive likelihood of each test input under the model. As GP produce a Gaussian predictive density, the log predictive likelihood is defined as

$$\log p(y_*|\mathcal{D}, \mathbf{x}_*) = -\frac{1}{2}\log(2\pi\sigma) - \frac{(y_* - f(\mathbf{x}_*))^2}{2\sigma^2} + \text{const}, \tag{6.1}$$

where $\sigma^2 = \sigma_*^2 + \sigma_n^2$, the sum of the predictive and noise variance respectively.

As we have trained each subset of data independently, we will compute the average of the RMSEs and MPLLs across all subsets, and use them as a comparison between the different prediction methods. The results of the various methods are presented in Table 6.1. From the results, it is apparent that our choice of using GP as our prediction method is ideal, as it gives the lowest RMSE and highest MPLL.

To determine the competitiveness of our model, we compare the error esti-

Prediction method	RMSE	MPLL
BLR	$1.02\pm1.34 \times 10^2$	-5.76±1.27
BLR*	$0.92\pm1.25 \times 10^2$	-5.61±1.10
RVM	$0.98\pm1.53 \times 10^2$	-6.47±1.17
GP	$0.83\pm1.63 \times 10^2$	-5.50±1.44

Table 6.1: Test results on various prediction methods on our dataset. In BLR*, interaction terms and terms raised to different powers were used, on top of linear terms, to capture possible effects of non-linear terms on the model.

mates of our prediction method against the published results of Zoopla's prediction model [10], by the proportion of predictions that fall within ±10% and ±20% of the actual selling prices. We randomly selected 1000 data points from the most recent price paid data in 2004 as our test set. As seen in Table 6.2, our prediction system has a slightly lower accuracy compared to Zoopla's prediction system. This is not particularly surprising as we utilise fewer variables in our prediction system compared to Zoopla. To improve on this, we would need to obtain more data and consider more factors that affect housing prices in our model, which is a major consideration for a future extension of this project.

Prediction system	Proportion of estimates	
	Within ±10%	Within ±20%
Zoopla	47.6%	75.0%
Ours	39.0%	62.0%

Table 6.2: A comparison on the proportion of estimates that falls within ±10% and ±20% of actual selling prices

6.2 Prediction System

To evaluate the performance of our prediction system, we record the time taken for the prediction system to generate the outputs for each query the user can make in our mobile application. 10 different predictions were made for each query at different locations (with a search radius of 1km for heatmap queries) and the minimum and maximum time taken are presented in Table 6.3.

Query	Minimum time taken	Maximum time taken
Prediction heatmap	13s	70s
Prediction graph	5s	10s
Budget heatmap	13s	80s

Table 6.3: Average time taken for each query in our prediction system to run

The heatmaps are expected to take a longer time than graphs to generate

due to the number of predictions required for the generation of heatmaps. Some locations will require a longer time than others to generate a prediction, as more data exists in the area which needs to be considered during computation.

Compared to an ideal situation where the user should receive the results within 0-3 seconds, the performance of our prediction system is poor. A closer look reveals that the bulk of the processing time comes from the number of predictions we are making. To render a smooth heatmap using the Google Maps Android API, around 1,200 evenly spread data points are needed for an area of radius 1km at Zoom Level 15 (a zoom level we feel is appropriate to clearly highlight the geographical variations of our predictions). The larger the search radius, the greater the number of data points are required.

Moreover, as GP is a non-parametric prediction method, the entire training data needs to be considered for making predictions. Recall that each prediction is made from the combination of predictions from the local area and surrounding areas. When more predictions are made for a single heatmap query, more training data needs to be loaded into the MATLAB environment. Thus, optimising the loading time of training data and reducing the number of predictions required are potential areas to explore in an extension of this project.

6.3 Mobile Application

To gather some feedback on the user experience and functionalities of our mobile application, we asked a few potential homebuyers to use our application. The feedback we received was very positive on the overall, with many stating it is a novel way to visualise the geographical variations in housing prices within London. Some have also commented that the application was easy to use and understand, which is a plus point in making the application accessible to all.

Nevertheless, we did receive some feedback that are less positive, where many commented on the time taken to generate the predictions. This has been mentioned in the section earlier and will be one of the main aspects of the application we need to improve on. Other criticisms include:

- Poor rendering of heatmap, resulting in some jagged edges after zooming in or out on the heatmap
- Inability to view predictions by particular boroughs or districts
- Unsure of the reliability of the predictions, as the predictions in certain regions appear counter-intuitive

Keeping these criticisms in mind, we can act upon them in the future to make our mobile application more accurate and accessible to all.

Chapter 7

Conclusion

We have managed to produce a mobile application that provides users with a novel way to look at future housing price predictions within London. Several regression methods have been explored and compared, before arriving at a prediction method based on GP for regression. Linear prior mean functions have been used in our model, so that future price predictions will tend towards more sensible values. We devised a way to utilise as much data as possible in our prediction system, by adopting the concepts of distributed GPs. Taking into account the performance of our application, we have separated our application into a client-side and server-side, such that heavy computations are done on the server-side and generation of visualisations are done on the client-side. The success of our approach to creating a mobile application for generating predictions can be applied to other problem sets concerned with geographical variations in the prediction model.

7.1 Future Work

Despite having produced a working application that met our initial requirements, there are various improvements that can be made in the future. These include improvements we did not make due to limited time on the project, and suggestions provided by users after using our mobile application.

- **Consider more factors affecting housing prices.** As discussed in Section 2, one main drawback of our prediction model is the lack of access to information. We would need to seek alternative sources of data besides the Land Registry, which can provide us with information on the area, the number of rooms, etc for each property. Economic factors such as the yearly inflation rate or GDP growth can also be considered.

- **Consider heteroscedastic GP regression [45].** Under the conventional GP regression method that we are using in our prediction method, the noise level is assumed to be uniform throughout the domain. However, this assumption may not be true. By considering a heteroscedastic treatment of noise, we can treat noise as a random variable. This can potentially better account for the difference in data points across the years in our dataset. An alternative noise model can also be explored, by training the data against the logarithm of the prices.

- **Optimise the prediction system through parallelised computations.** As discussed in Section 5.2, a major concern with the prediction system is the loading time. Moreover, our dataset takes more than one day to train. Rather than performing the computations sequentially, we can use multiple processors and parallelise the computations involved, which can potentially reduce the training time and prediction time. Another way to approach this problem is to look for alternative APIs that allow us to produce heatmaps of similar quality but require fewer data points, or faster libraries to implement GPs.

- **Add more functionalities into the application.** As mentioned by one of the users, we can provide options for user to select a borough or district to generate the heatmaps, instead of entering a postcode and selecting a search radius. We can also look into ways to generate a heatmap for the whole of London initially in a zoomed-out view, only requesting for more data points when the user zooms in to display a clearer picture.

Bibliography

[1] GOV.UK. (2013) Property transactions in the UK. [Online]. Available: https://www.gov.uk/government/collections/property-transactions-in-the-uk

[2] London Datastore. (2015) London Housing Market Report. [Online]. Available: http://data.london.gov.uk/housingmarket/

[3] Nationwide. (2015) Nationwide House Price Index. [Online]. Available: http://www.nationwide.co.uk/~/media/MainSite/documents/about/house-price-index/Q1_2015.pdf

[4] CEBR. (2015) UK house prices to decline this year, dragged down by a slump in London. [Online]. Available: http://www.cebr.com/reports/uk-house-prices-to-fall/

[5] A. White. (2015) London house prices to fall up to 5pc as sellers abandon the market. [Online]. Available: http://www.telegraph.co.uk/finance/property/11631377/London-asking-prices-jump-17pc-after-the-election.html

[6] ——. (2015) London asking prices jump 17pc after the election. [Online]. Available: http://www.telegraph.co.uk/finance/property/11631377/London-asking-prices-jump-17pc-after-the-election.html

[7] Rightmove. (2014) House prices to rise 30pc by 2019, new Rightmove and Oxford Economics forecast reveals. [Online]. Available: http://www.rightmove.co.uk/news/articles/property-news/house-prices-to-rise-30-by-2019-new-rightmove-and-oxford-economics-forecast-reveals

[8] ——. (2015) House Price Index. [Online]. Available: http://www.rightmove.co.uk/news/house-price-index

[9] Savills PLC. (2014) House Price Predictions. [Online]. Available: http://www.savills.co.uk/resources/5-year-forecast/

[10] Zoopla. (2015) About our value estimates. [Online]. Available: http://www.zoopla.co.uk/property/estimate/about/

[11] C. M. Bishop, *Pattern Recognition and Machine Learning.* Springer, 2006.

[12] K. P. Murphy, *Machine Learning: A Probabilistic Perspective.* MIT Press, 2012.

[13] S. J. Sheather, "Density Estimation," *Statistical Science*, vol. 19, no. 4, pp. 588–597, 2004.

[14] B. W. Silverman, *Density Estimation for Statistics and Data Analysis.* CRC Press, 1986.

[15] L. van der Maaten, E. Postma, and H. van den Herik, "Dimensionality Reduction: A Comparative Review," *Elsevier*, 2008.

[16] C. J. C. Burges, *Dimension Reduction: A Guided Tour.* Now Publishers Inc, 2010.

[17] A. Jain, M. Murty, and P. Flynn, "Data Clustering: A Review," 2009.

[18] P. Dayan and C. J. Watkins, "Reinforcement Learning," in *Encyclopedia of Cognitive Science.* MacMillan Press, 2001.

[19] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis.* John Wiley and Sons, 2012.

[20] I. J. Myung, "Tutorial on maximum likelihood estimation," *Elsevier Science*, 2003.

[21] D. Lindley and A. Smith, "Bayes Estimates for the Linear Model," *Journal of the Royal Statistical Society*, vol. 34, no. 1, pp. 1–41, 1972.

[22] M. E. Tipping, "The Relevance Vector Machine," in *Advances in Neural Information Processing Systems*, vol. 12, 2000, pp. 652–658.

[23] ——, "Sparse Bayesian Learning and the Relevance Vector Machine," *Journal of Machine Learning Research*, vol. 1, pp. 211–244, 2001.

[24] B. Schölkopf and A. J. Smola, *Learning with Kernels.* MIT Press, 2002.

[25] B. Schiilkopf, "The kernel trick for distances," in *Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference. Vol. 13.* MIT Press, 2001.

[26] C. J. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery 2*, pp. 121–167, 1998.

[27] C. Cortes and V. Vapnik, "Support-Vector Networks," in *Machine Learning*, 1995, pp. 273–297.

[28] J. Shawe-Taylor and N. Cristianini, *Support Vector Machines and other kernel-based learning methods.* Cambridge University Press, 2000.

[29] M. E. Tipping and A. C. Faul, "Fast Marginal Likelihood Maximisation for Fast Marginal Likelihood Maximisation for Sparse Bayesian Models," in *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, 2003.

[30] J. Quinonero-Candela, "Learning with uncertainty - Gaussian processes and relevance vector machines," Ph.D. dissertation, Technical University of Denmark, 2004.

[31] C. E. Rasmussen and J. Quinonero-Candela, "Healing the Relevance Vector Machine through Augmentation," in *Proceedings of the 22nd International Conference on Machine Learning*, 2005.

[32] C. K. I. Williams and C. E. Rasmussen, "Gaussian Processes For Regression," in *Advances in Neural Information Processing Systems*, vol. 8, 1996, pp. 514–520.

[33] J. Hensman, N. Fusi, and N. D. Lawrence, "Gaussian Processes for Big Data," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2013.

[34] C. E. Rasmussen, "Evaluation of Gaussian processes and other methods for non-linear regression," Ph.D. dissertation, University of Toronto, 1996.

[35] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning.* MIT Press, 2006.

[36] M. P. Deisenroth and J. W. Ng, "Distributed Gaussian Processes," in *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

[37] J. Quinonero-Candela and C. E. Rasmussen, "A Unifying View of Sparse Approximate Gaussian Process Regression," *Journal of Machine Learning Research*, vol. 6, pp. 1939–1960, 2005.

[38] J. Quinonero-Candela, C. E. Rasmussen, and C. K. Williams, "Approximation Methods for Gaussian Process Regression," *Large-scale kernel machines*, pp. 203–223, 2007.

[39] J. W. Ng and M. P. Deisenroth, "Hierarchical Mixture-of-Experts Model for Large-Scale Gaussian Process Regression," 2014.

[40] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive Mixtures of Local Experts," *Neural Computation*, vol. 3, no. 79-87, 1991.

[41] C. E. Rasmussen and Z. Ghahramani, "Infinite mixtures of gaussian process experts," *Advances in Neural Information Processing Systems*, no. 881-888, 2002.

[42] Y. Cao and D. J. Fleet, "Generalized Product of Experts for Automatic and Principled Fusion of Gaussian Process Predictions," 2014.

[43] H. Alkharusi, "Categorical Variables in Regression Analysis: A Comparison of Dummy and Effect Coding," *International Journal of Education*, 2012.

[44] IDC. (2015) Smartphone OS Market Share, Q1 2015. [Online]. Available: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[45] A. J. S. Quoc V. Le and S. Canu, "Heteroscedastic Gaussian Process Regression," in *Proceedings of the 22 nd International Conference Proceedings of the 22nd International Conference on Machine Learning*, 2005.