University of Glasgow | School of Computing Science

Honours Individual Project Dissertation

Applications of RNA Velocity and Gaussian Processes for scRNA-seq Data

Inna Ivanova

June 17, 2020

Abstract

Single cell RNA-sequencing (scRNA-seq) has opened up the possibilities of the in-depth studying of individual cells in complex biology systems. A novel way for gene profiling is proposed in the form of unsupervised ranking of genes by smoothness. The use of Gaussian Processes to identify smooth genes is described and augmented with the inclusion of RNA Velocity. The identification of such genes is of particular interest as they can describe some properties of the biological system and unravel hidden structures in the dataset. RNA Velocity is demonstrated to result in smoother fit for most genes and generally better results, however, introduced noise can also be present. The proposed implementation runs 150 times faster than alternatives and is compatible with existing tool sets.

Acknowledgements

I would like to highlight my deepest gratitude to my supervisor, Dr. Simon Rogers, for taking up this project and for his immense support, timely feedback and unparalleled assistance which were paramount for the completion of this project.

Education Use Consent

I hereby grant my permission for this project to be stored, distributed and shown to other University of Glasgow students and staff for educational purposes. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Signature: Inna Ivanova Date: 6 April 2020

Contents

1 Introduction

This dissertation presents process of researching a new method for gene profiling, with focus on ranking of genes by their smoothness. The dissertation documents the critical analysis of the problem and details of the proposed solution, thus giving an insight into the research results, technical achievements, limitations of the solution and future improvements.

1.1 General Overview

In recent developments in the field of of high-throughput biology, it has become possible to analyse gene expression (the number of reads that map to each gene (read count) in a cell) of thousands of genes across thousands of individual cells over time. This new technology is called single cell RNA-sequencing (scRNA-seq) and was chosen as the breakthrough technology of the year by the scientific journal 'Science' in 2018. scRNA-seq has opened up the possibilities of the in-depth studying of individual cells in complex biology systems and revealing a selection of gene candidates that could exhibit interesting properties.

scRNA-seq dataset are derived by processing very long sequences of RNA molecules, which result in gigabytes of raw textual data per cell sample. Managing scRNA-seq datasets is a computational challenge which is becoming more important due to the tendency of increase in dataset size year over year. scRNA-seq is therefore strongly dependent on the developments in computer science at all stages of the analysis process from filtering raw data to final visualization. This includes algorithms for string alignment, dimensionality reduction, unsupervized feature identification (clustering), modeling cell differentiation and others.

The latest development in modeling celluar differentiation is RNA Velocity as proposed by La Manno (2018). This technology allows the prediction of future state of individual cells on the timescale of hours. This could potentially contribute to the analysis of celluar dynamics and developmental lineages.

1.2 General Problem

In this dissertation the focus will be specifically on the development of new method for gene profiling. A novel way for unsupervised ranking of genes using Gaussian Process to identify smooth genes in the low dimensional mapping (2D) of the dataset is described. The discovery of such genes is of particular interest as they can describe some properties of the biological system and unravel hidden structures in the dataset. In addition, EL Low (2019) suggests that smoothly varying genes have been identified as possibly relating to consequence of differentiating events.

The proposed Gaussian Process model has several advantages. The definition of the smoothness is well described by the properties of the kernel function of the Gaussian Process. The likelihood of the Gaussian Process can be directly used to optimize the models hyperparameters. Furthermore, it can be used as a criteria for the smoothness of the gene. Finally, Gaussian Processes can incorporate derivative observations information. As a result, RNA Velocity can be included in the model in the form of derivatives at the observation points.

1.3 Outline

The rest of the dissertation is structured as follows. Chapter Background presents the background of the case study, describing the general concepts of single cell RNA Sequencing, different RNA Velocity models, popular algorithms for dimensionality reductions and Gaussian Processes. Chapter Analysis explains the problem domain, describes a proposed solution, gives an insight of the alternatives and discusses the problems regarding the proposed solution. Chapter Design summarizes the pipeline/the flow of the. Chapter Implementation briefly explains the current state of the project, details of the implementation and the technical achievements. Chapter Evaluation provides a detailed discussion on the performance of the proposed solution, final results and evaluation of different models used for the problem. Chapter Conclusion summarizes the project and gives insight on possible future improvements.

2 Background

This chapter gives insight of the single cell RNA sequencing (scRNA-seq for short) approach, different scRNA-seq protocols, analysis methods and processing of scRNA-seq data and its applications, introduction to Gaussian Processes and overview of dimensionality reduction algorithms.

2.1 Single Cell RNA Sequencing

Single cell RNA sequencing is a new technology for the deep sequencing of individual RNA and DNA molecules. This technology measures the distribution of gene expression levels across a population of cells. Compared to bulk RNA-seq approach, which only looks at the average gene expression across a large population of cells, scRNA-seq is able to catch transcriptional differences and similarities across a population of cells as it's looking at the gene expression levels for each individual cell. This is very useful for heterogeneous systems which includes early development of cells and complex tissues like the brain. scRNA has

The method was first introduced by Tang (2009) in 2009 and it has ever since been developed with new protocols and cost-efficient sequencing methods. The resulting datasets vary in range from 10^2 to 10^6 cells in a single dataset. In recent years computational challenges have been increasing as datasets are becoming bigger in size. Nevertheless, it is continuing to grow in popularity due to its numerous advantages. It allows for the analysis of the transcriptomes of single cells providing higher resolution of cellular difference, better understanding of the function of an individual cell and others described by Eberwine (2014).

2.1.1 scRNA-seq Workflow

Visual representation of the process is shown in Figure 2.1. The first step in conducting the experiment is isolating the cells of interest from the tissue. Each cell is then captured by a single micro-fluid droplet or sorted by laser ablation into a different well. The cells are then lysed in order to capture as many RNA molecules as possible. These molecules are then reverse transcribed from RNA to cDNA. The amount of available RNA molecules is low so they are amplified in order to produce more data from each cell. The amplified RNA or cDNA molecules are then sequenced. The resulting sequences and their quality scores are stored in a text-based formats specifically developed for biological sequenced data. Then with the help of sequence alignment tools the raw sequence reads are aligned with known gene sequences and the result of the experiment is summarized into a big count matrix. The matrix rows identify individual cells and the columns identify genes found in all the cells. The matrix values are the counts of each gene within the cell (gene expression). Further analysis can be conducted on the raw count matrix in order to extract meaningful information for the cells and the genes.

Common issues with the scRNA-seq dataset described by Parekh (2016) are the levels of drop-outs[1] and the noise and bias introduced during this amplification. If the capture of individual cells is not successful the reads might not contain the full transcriptomics of the cell. This could lead

[1] Drop-outs are defined as the phenomena where a gene is found at a low or moderate intensity level in one cell but is absent in another cell of the same type.

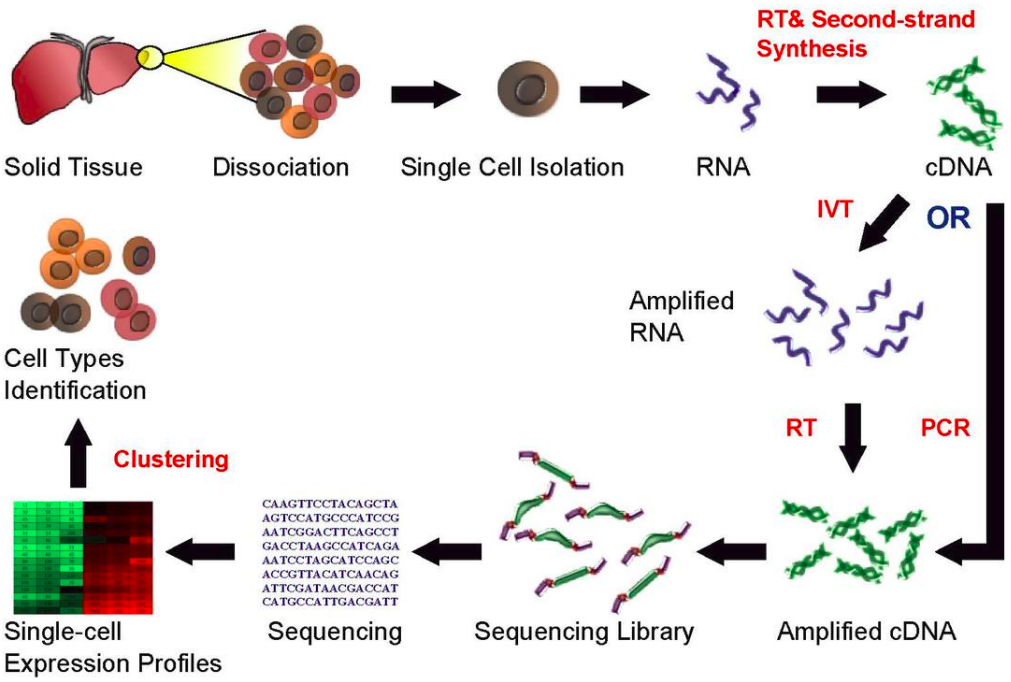


Figure 2.1: *Single Cell RNA Sequencing Workflow. Sourced from https://en.wikipedia.org/wiki/RNA-Seq*

to many missing values and confusion whether the reads are property of the cell or a drop-out. For example, some genes are found in thousands of cells, others in hundreds of cells. If the reads in a cell seem too sparse this could mean either a drop-out event or property of the specific cell. Furthermore, the amplification process amplifies these drop-outs as well as introduces fragmentation bias and noise to the dataset. Methods on how these issues are being dealt with are described in subsection 2.1.3.

2.1.2 scRNA-seq Protocols

There are hundreds of scRNA-seq methods with the most popular being Smart-Seq2 and 10X Genomics. In this dissertation datasets of interest have been obtained with 10X Genomics protocol. The pros and cons of different protocols are stated in Table 2.1.

2.1.3 scRNA-seq Analysis Methods

As described in subsection 2.1.1 the result of the experiment is summarized into a count matrix of gene occurrences (columns) in each cell (rows). Each count identifies the gene intensity in the specific cell. As there are tens of thousands of genes and tens of thousands of cells, it becomes computationally challenging to manipulate and analyze the whole dataset. In addition, many cells might have been corrupted during the experiment so the quality of the reads and the cells might be low. Thus, filtering of the dataset and suitable visualisation techniques are necessary.

A popular toolkit specifically build for the analysis of single-cell gene expression datasets initially developed by Wolf (2018) is scanpy. It is an open-sourced project for scientific analysis with a Python API. Its build-in functionality includes numerous analysis methods for preprocessing, transformation and visualisation of the dataset.

There are standard steps in the processing of the raw count matrix. First, outliers such as genes

Protocol	Method	Pros	Cons
Smart-Seq2	Well-based	1. Very little starting material necessary; 2. Improved coverage across transcripts; 3. High level of mappable reads.	1. Preferential amplification over high-abundance transcripts; 2. Transcript length bias, 3. Inefficient transcription of reads over 4 Kb; 3. Purification step may lead to loss of material;
10XGenomics	Droplet-based	1. High throughput single-cell transcriptome profiling; 2. Highly scalable to larger cell quantities; 3. Cell capture rate up to 65% .	1. Droplets may contain two cells or two different types of barcodes. 2. mRNA noise is higher in the low expression levels

Table 2.1: *scRNA-seq Protocols with specified method and stated pros and cons for each protocol. Based on analysis from https://emea.illumina.com/ and https://www.10xgenomics.com/.*

expressed in small number of cells and cells with small number of genes need to be filtered. Then the count matrix is normalized by the total number of counts per cell so that gene expressions become comparable across all cells. As gene counts are exponentially distributed the logarithm of the normalized matrix is preferred for further analysis. Additional filtering based on highest-variable genes and number of counts can be conducted. Finally, gene profiling and hidden structures can be obtained with the help of dimensionality reduction algorithms and clustering analysis. The pipeline is shown in Figure 2.2.

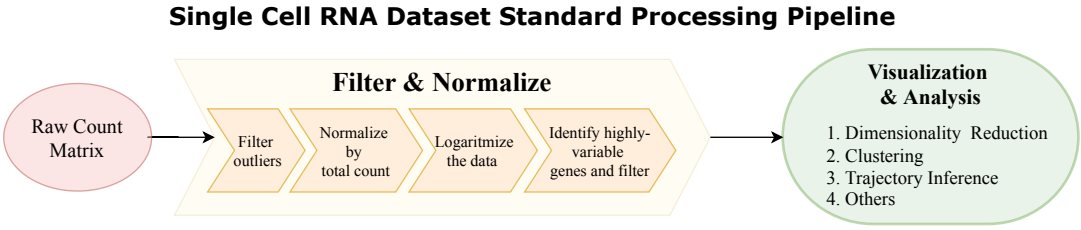


Figure 2.2: *A standard pipeline for processing scRNA-seq dataset. All the needed steps are included as methods in the* `scanpy` *library.*

2.2 Dimensionality Reduction Algorithms

Dimensionality reduction is the process of reducing the number of random variables in a system by obtaining a set of principal variable components. Dimensionality reduction algorithms are particularly useful for visualization of high dimensional data and can unravel hidden structures in the dataset. They are necessary step in the processing of scRNA-seq datasets because this type of datasets are extremely high dimensional (thousands of dimensions) and sometimes very sparse, and therefore need to be presented in a meaningful and readable manner. There are many standard dimensionality reduction algorithms which are commonly used for different high dimensional datasets. The three main algorithms used in this dissertation are Principal Component Analysis, t-distributed Stochastic Neighbor Embedding and Uniform Manifold Approximation and Projection.

2.2.1 Principal Component Analysis

Principal Component Analysis (PCA) is a statistical method that uses an orthogonal transformation of the data in order to transform it into linearly independent sets of values called principal components. This transformation is defined in such a way that the principal components are ranked by their importance. The main idea of PCA is to reduce the dimensions of the given data but preserve as much as possible of the variance therefore keeping a good representation of the data with only using a subset of the dimensions.

2.2.2 T-distributed Stochastic Neighbor Embedding

The T-distributed Stochastic Neighbor Embedding (t-SNE) is an unsupervised machine learning algorithm for nonlinear dimensionality reduction. It is based on the distribution of the datapoints. The algorithm computes the probability distribution of pairs of datapoints in the original high dimensional space in such a way that similar (closely related) objects have high probability of being picked up. Then, the algorithm produces similar probability distribution in a low dimensional map and minimizes the difference in the distributions of the high and low dimensional space. This results in a low dimensional embedding that produces the same distribution as the high dimensional data. It is commonly used for both clustering analysis and visualisation of scRNA-seq datasets.

2.2.3 Uniform Manifold Approximation and Projection

The Uniform Manifold Approximation and Projection algorithm for Dimensionality Reduction (UMAP) proposed by Leland McInnes (2018) is a nonlinear dimensionality reduction technique. The algorithm for UMAP is very similar to the one of t-SNE. The major difference between the two, however, is the way the similarity measure is defined. While t-SNE preserves only the local structure in the data, UMAP is generally considered to preserve both local and global structure in the data. UMAP performance is comparable to t-SNE and the algorithm is especially popular with clustering analysis. It is considered state-of-the-art for visualization of scRNA-seq datasets.

2.3 RNA Velocity

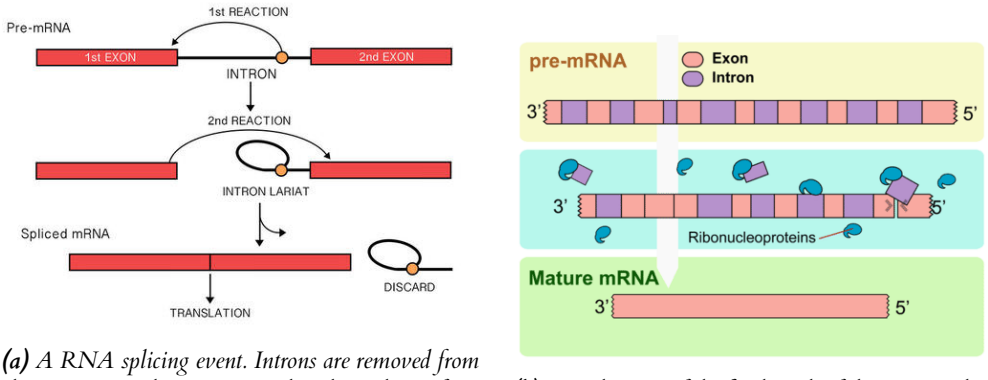
When conducting the scRNA-seq experiment the measurements end the cells' life. Thus, scRNA-seq datasets provide only a static snapshot at a point in time of the cells of interest. However, being able to derive conclusions for the further development of cells is very important in the understanding of heterogeneous systems, decease development, tissue regeneration, identification of driver genes and other complex biological processes.

The recently proposed idea by La Manno (2018) of RNA Velocity introduced a new technique of the in-depth studying of cellular differentiation. RNA Velocity is defined as the time derivative of the gene expression state and can be directly computed from the ratio of observed spliced and unspliced mRNA molecules.

2.3.1 RNA Splicing

RNA splicing is a RNA process in which newly made precursor mRNA (pre-mRNA) transcript is transformed into mature mRNA molecule. In this spicing process the pre-mRNA transcript undergoes a removal of intronic sequences resulting in exonic transcript sequences which are part of the encoding of the final mature mRNA. See Figure 2.3.

Unspliced transcripts are generally unstable and are rapidly cleaned from the cell during the RNA splicing event. The intron sequences which are saved in the DNA encoding are not present



(a) *A RNA splicing event. Introns are removed from the transcript and exons are combined together to form the final matured mRNA.*

(b) *Visualisation of the final result of the RNA Splicing.*

Figure 2.3: *The RNA Splicing event and the resulting matured mRNA molecules. Sourced from https://www.ck12.org/biology/rna-types/lesson/Messenger-RNA-Advanced-BIO-ADV/*

in mature mRNA transcript from which protein is made. This means the RNA sequences for specific tissue do not include intron sequences which are generally important for the control of the gene expressions.

2.3.2 RNA Velocity Steady Model

La Manno (2018) originally proposed the RNA Velocity steady-state model which is defined as the observed ratio of spliced and unspliced mRNA from an inferred steady state. The model is formed by the differential Equation 2.3

$$\frac{ds(t)}{dt} = \beta u(t) - \gamma s(t) \tag{2.1}$$

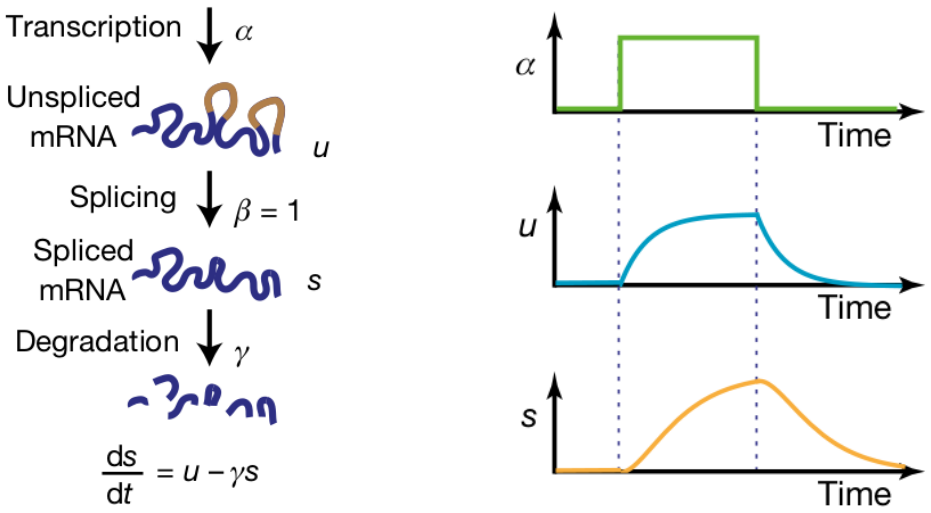
where $s(t)$ is the amount of spliced mRNA molecules after time t, $u(t)$ is the amount of uspliced mRNA molecules after time t, β is the splicing rate of the mRNA molecules and γ is the degradation rate of the mRNA molecules.

There are two assumptions this model enforces: the splicing rate β is a constant across all cells ($\beta = 1$); the mRNA levels reach a steady-state equilibrium. The steady-state equilibrium is approached asymptotically when the rate of transcription α is constant. This means $\frac{ds(t)}{dt} = 0$ and there is a fixed-slope relation $u(t) = \gamma s(t)$ resulting in a solution for γ. See Figure 2.4. The resulting RNA Velocity is a high dimensional vector and can be projected in the low level embedding of the original scRNA-seq data.

2.3.3 RNA Velocity Dynamic Model

The assumptions introduced in the original RNA Velocity framework are often violated in dynamical system like cell differentiation. Thus, RNA Velocity Dynamic Model that solves the full transcriptomics dynamics of the system was introduced by Volker Bergen (2019) as a generalization of the steady-state model.

The original RNA Velocity framework assumes several conditions which are not necessarily true: splicing rate across all cells is a constant, all the cells reach a steady state. This is taken into account in the introduced dynamic model. The full transcriptomics equations are solved independently for each gene making the velocity more accurate and comparable across genes.



(a) Visualization of the transcription of RNA along with splicing and degradation.

(b) Relationship between the transcription rate α and the spliced and unspliced amount of RNA.

Figure 2.4: *The transcription of RNA and the relationship between transcription rate α and the abundance of RNA. Taken from La Manno (2018)*

.

The full transcriptional dynamics are defined as

$$\frac{du(t)}{dt} = \alpha_k(t) - \beta u(t) \tag{2.2}$$

$$\frac{ds(t)}{dt} = \beta u(t) - \gamma s(t) \tag{2.3}$$

where $\alpha_k()$ is the reaction rates of transcription.

The details of the of solving these equations are explained in the Dynamical model sections of the paper Volker Bergen (2019). The resulting high dimensional RNA velocity vector can still be projected into the low dimensional embedding of the scRNA-seq data.

2.4 Gaussian Process

Gaussian processes are very powerful tool for both regression and classification tasks and are an example of a Bayesian approach model where predictions are made based on some observations in the past. While classical Bayesian Regression is a probabilistic approach to predicting distributions over set of variables, Gaussian Processes take that one step further and predict distributions over set of functions with continuous domain. Gaussian Processes are non-parametric models as they have infinitely many variables. The distribution of a Gaussian Process is the joint distribution of all those infinitely many random variables. Furthermore, Gaussian Processes can adopt additional information for the derivatives of the input datapoints and constrain its distribution even further. Gaussian Processes are very good for capturing non-linear relationships and can further benefit from GPUs and TPUs as the computations needed are matrix operations.

2.4.1 Multinomial Distribution, Covariance Matrix and Conditioning

Essential parts of the Gaussian process model are the multinomial Gaussian distribution, its covariance matrix and the concept of conditioning. The multinomial Gaussian distribution is a generalization over the univariate Gaussian distribution. It presents the relationship between pairs of variables and it is defined as in Equation 2.4 and Equation 2.5.

$$\boldsymbol{x} = (x_1, x_2, ..., x_k)^T \tag{2.4}$$

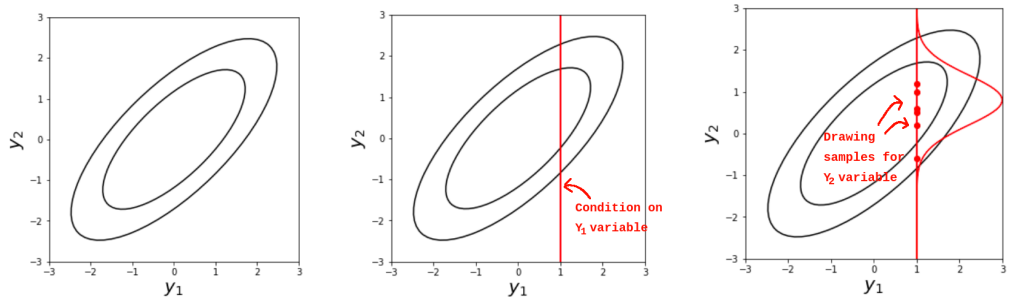
$$\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma) \tag{2.5}$$

Where $\boldsymbol{x}$ is the random vector drawn from the k-dimensional Gaussian distribution, $\boldsymbol{\mu}$ is the mean vector and Σ is the covariance matrix.

There are two main operations on a Gaussian distribution are relevant to the discussion. These are:

1. Sample random variables (vectors) from the distribution.
2. Condition on a set of variables and make predictions for other random variables based on the initial conditioning.

Conditioning on a variable for a value is essentially taking a slice through the multivariate Gaussian distribution. By the properties of the multivariate Gaussian distribution this slice is another Gaussian distribution. The new Gaussian distribution can be used for constrained predictions on other variables based on the initial condition. A simple example for variables from binomial Gaussian distribution of this operation is shown in Figure 2.5.



(a) Multinomial ditribution for variables y_1 and y_2.

(b) Condition on y_1 variable for specific value.

(c) Drawing samples for y_2 given condition on y_1.

Figure 2.5: *Making prediction for variable y_2 given condition on variable y_1. (a) shows the distribution of variables y_1 and y_2. (b) shows how conditioning at specific value for variable y_1 results at a well-defined distribution over variable y_2 (c) shows drawing samples from the distribution over variable y_2 when condition on the y_1 variable is applied.*

The sampling and conditioning as shown in Figure 2.5 can be applied for higher dimensions. To visualise the effect of sampling in higher dimensions another approach is used. The index of the variables and their sampled values are used as the main 2 axes on the grid and the resulting data points are connected. By increasing the number of dimensions the resulting data points gradually span a continuous space and describe continuous functions. When extending to infinite dimensions, curves drawn from the infinite dimensional Gaussian distribution describe the whole space of possible functions. The visualization of this approach is shown in Figure 2.6.

This approach for expanding into infinite dimensions can also be applied when the distributions are conditioned by a set of variables. This is shown in Figure 2.7.

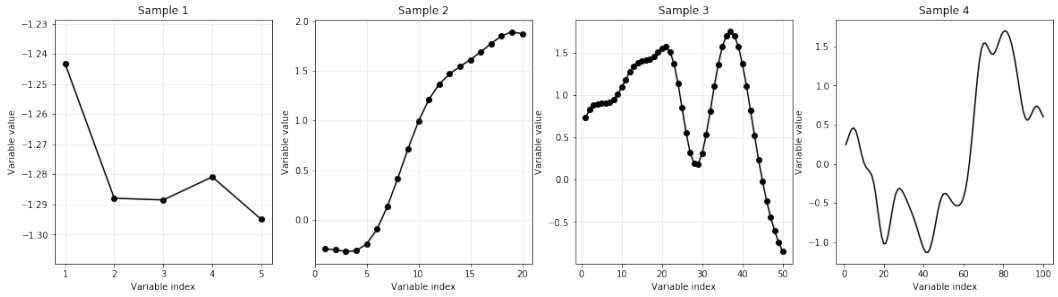


Figure 2.6: *Drawing random samples from Gaussian Distributions can be extended to continuous space.*

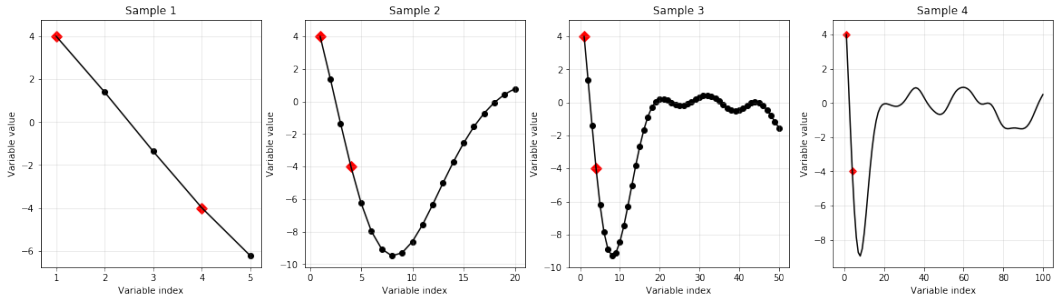


Figure 2.7: *Drawing random samples from Gaussian Distributions while conditioning on the first and fourth variable.*

Sampling many times from the same infinite dimensional Gaussian distribution when conditioning on a set of variables is applied results in different function samples that pass through the conditioned variables. This is shown in Figure 2.8.

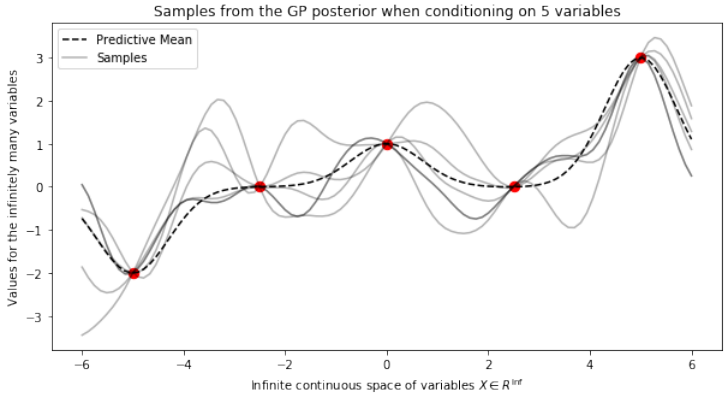


Figure 2.8: *Drawing random samples multiple times from infinite Gaussian Distribution while conditioning on 5 variables.*

The conditioning example in Figure 2.8 results in a family of curves that fit the initial observations. The mean and the standard deviation can be obtained analytically (see subsection 2.4.2) so there is no need for large amount of samples. The sampling and conditioning can also be applied for input variables in higher dimensions ($\boldsymbol{x} \in R^D, D \geq 1$) resulting in not only functions but also surfaces.

To summarize, the process of conditioning on a set of variables followed by sampling from a

distribution to predict for unknown data is essentially the core principle of how Gaussian Processes are operating. As there are infinite number of parameters ($x_1, x_2, ..., x_\infty$), the Gaussian Process is a non-parametric model which is entirely defined by its mean and covariance. The covariance matrix of a Gaussian process is defined by a function called the kernel (see subsection 2.4.3) that specifies the similarity/correlation of the variables (train and test data).The mathematics and the formal definitions behind the Gaussian Process are explained in subsection 2.4.2.

2.4.2 Formal Definitions and Mathematics Behind Gaussian Process

The formal definition of a Gaussian Process is a collection of random variables, any finite number of which have consistent Gaussian distributions. Gaussian Processes are entirely defined by a mean function $m(\boldsymbol{x})$ and a covariance function $K(\boldsymbol{x}, \boldsymbol{x}')$.

Assume the system of interest f is defined as

$$\boldsymbol{y} = f(\boldsymbol{X}) + v \tag{2.6}$$

where $\boldsymbol{X} = [\boldsymbol{x}^1, \ldots, \boldsymbol{x}^M], \boldsymbol{x} \in R^D$ are the initial observations, $\boldsymbol{y} \in R$ are the true values and v is the additive noise in the measurements. To describe the whole system by a Gaussian process the noise v is assumed a Gaussian noise $v \sim \mathcal{N}(0, \sigma^2)$.

The underlying system f is modeled by a Gaussian process as

$$f \sim \mathcal{GP}(m(\boldsymbol{x}), K(\boldsymbol{x}, \boldsymbol{x}')) \tag{2.7}$$

In the general case, there are some points $\boldsymbol{X}$ for which the output f is known and there are other points $\boldsymbol{X}_*$ for which the output f_* needs to be obtained based on the already known observations from $\boldsymbol{X}$ and f. This is achieved by finding the conditional probability $p(f_*|\boldsymbol{X}_*, \boldsymbol{X}, f)$ assuming that f and f_* are jointly Gaussian.

The posterior of the Gaussian process is the joint probability of the outcome values and is defined as

$$\begin{pmatrix} f \\ f_* \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu}_* \end{pmatrix}, \begin{pmatrix} K(\boldsymbol{X}, \boldsymbol{X}) & K(\boldsymbol{X}_*, \boldsymbol{X}) \\ K(\boldsymbol{X}, \boldsymbol{X}_*) & K(\boldsymbol{X}_*, \boldsymbol{X}_*) \end{pmatrix}\right) \tag{2.8}$$

The predictive distribution f_* for a test case at $\boldsymbol{X}_*$ is given by averaging over the output of all the possible models with respect to the Gaussian posterior. It can be shown (please refer to Rasmussen and Williams (2006)) that the distribution of f_* is then described by another Gaussian as

$$f_* \sim \mathcal{N}(\boldsymbol{\mu}_*, \Sigma_*) \tag{2.9}$$

where the posterior mean $\boldsymbol{\mu}_*$ is defined as

$$\boldsymbol{\mu}_* = K(\boldsymbol{X}_*, \boldsymbol{X})K(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{y} \tag{2.10}$$

the posterior covariance matrix Σ_* is defined as

$$\Sigma_* = K(\boldsymbol{X}_*, \boldsymbol{X}_*) - K(\boldsymbol{X}_*, \boldsymbol{X})K(\boldsymbol{X}, \boldsymbol{X})^{-1}K(\boldsymbol{X}, \boldsymbol{X}_*) \tag{2.11}$$

where $\boldsymbol{X}$ are the train points (initial observations) and $\boldsymbol{X}_*$ are the test points.

To make prediction for unobserved data it is only needed to substitute the right observations in Equation 2.10. To draw random samples from the distribution of $f_* \sim \mathcal{N}(\boldsymbol{\mu}_*, \Sigma_*)$ it is only needed to compute the mean as in Equation 2.10 and covariance matrix in Equation 2.11.

For detailed derivations of Equation 2.10 and Equation 2.11 refer to the book by Rasmussen and Williams (2006).

2.4.3 Kernels and Hyperparameters

The kernel is a function that is a linear mapping of the input space which defines the covariance matrix and determines almost all the generalization properties of a Gaussian process model. Kernels measure the similarity between points and give different relationships between the random variables resulting in different set of family curves. The selection of the proper kernel function is very important for the given dataset and problem definition.

There are numerous kernel functions with the most popular being the Squared Exponent Kernel also known as the Radial Basis Kernel and is defined as

$$K(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \exp\left(-\frac{1}{2l^2}\|\boldsymbol{x} - \boldsymbol{x}'\|^2\right) \tag{2.12}$$

where the norm $\|\boldsymbol{x} - \boldsymbol{x}'\|$ is the Euclidean distance of the vectors $\boldsymbol{x}$ and $\boldsymbol{x}'$, σ^2 is the overall variance also known as the amplitude and l is the length-scale parameter.

Other common kernels are the Linear Kernel, the Rational Quadratic Kernel, the Matern Kernel and the Periodic Kernel. The effect on the family of curves generated by different kernels is shown in Figure 2.9

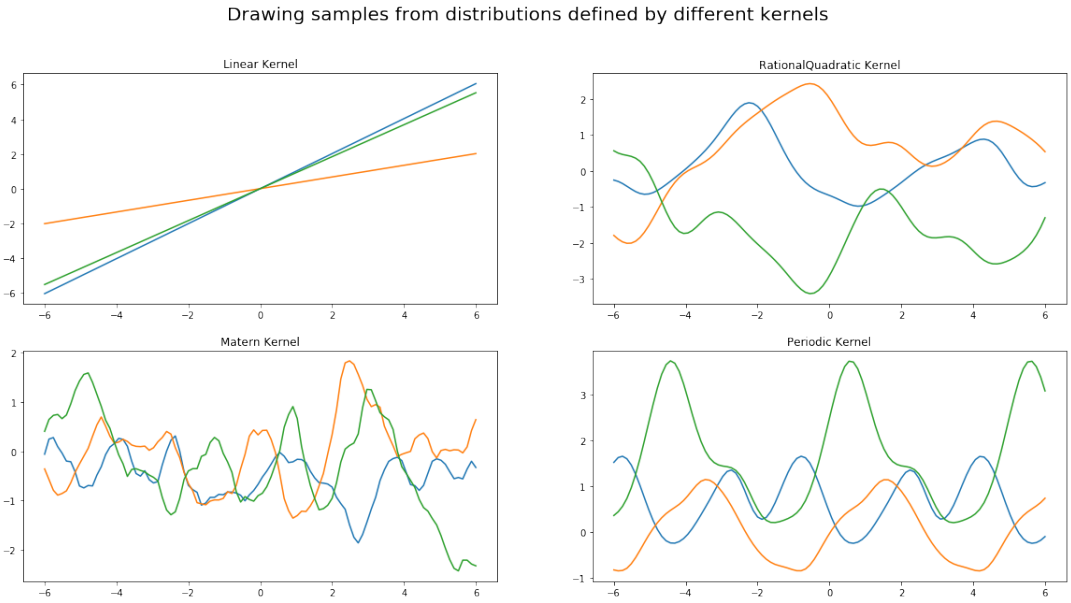


Figure 2.9: *The family of curves generated by different kernel functions.*

Kernels are linear mappings so they can be combined altogether (multiplication or summation, or both).

Most kernels have hyperparameters that need to be tuned. These hyperparameters define properties of the resulting curve families like smoothness, periodicity and others. For the Squared Exponent Kernel these hyperparameters are the length-scale parameter l and the amplitude σ^2. Choosing appropriate hyperparameters for the specific dataset and problem domain is very important otherwise the Gaussian process model can become too complex or too simple. The hyperparameters can be fine-tuned by optimizing the log marginal likelihood value and making use of its derivative.

2.4.4 Marginal Likelihood

The marginal likelihood is an essential metric for any Bayesian approach algorithm. It is called marginal likelihood as it is marginalizating a set of parameters that control the process. As the Gaussian process is non-parametric model it may be confusing to which are the parameters of the model. As mentioned in the above sections the Gaussian process is defined by a mean function and covavriance matrix. The covariance matrix is defined by a kernel function which comprises hyperparameters. As these hyperparameters define properties of the kernel function, they are the parameters controling the Gaussian process model complexity and quality.

Marginal likelihoods are generally hard to compute, however, in a Gaussian process the log marginal likelihood is well-defined. From Rasmussen and Williams (2006)

$$\log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) = -\frac{1}{2}\boldsymbol{y}^T K(\boldsymbol{X},\boldsymbol{X})^{-1}\boldsymbol{y} - \frac{1}{2}\log|K(\boldsymbol{X},\boldsymbol{X})| - \frac{n}{2}\log 2\pi \tag{2.13}$$

where $\boldsymbol{y}$ are the true values for the $\boldsymbol{X}$ initial observations, K is the covariance function (kernel), $\boldsymbol{\theta}$ are the hyperparameters of the covariance function and n is the number of observations $\boldsymbol{X}$.

The marginal likelihood is essentially the coefficient of the stochastic complexity. Thus, it is useful when assessing the model fit and when finetuning the hyperparameters of the kernel function. The logarithm of the marginal likelihood is usually used for numerical stability. The derivative of the log marginal likelihood for Gaussian processes is also well-defined and can be used in the optimization of the hyperparameters.

$$\frac{\partial}{\partial\theta_i}\log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) = \frac{1}{2}\boldsymbol{y}^T K(\boldsymbol{X},\boldsymbol{X})^{-1}\frac{\partial K(\boldsymbol{X},\boldsymbol{X})}{\partial\theta_i}K(\boldsymbol{X},\boldsymbol{X})^{-1}\boldsymbol{y} - \frac{1}{2}tr\left(K(\boldsymbol{X},\boldsymbol{X})^{-1}\frac{\partial K(\boldsymbol{X},\boldsymbol{X})}{\partial\theta_i}\right) \tag{2.14}$$

2.4.5 Gaussian Process with Derivative Observations

The derivative of a Gaussian process is another Gaussian process by the definition of the Gaussian process. Thus, information about the derivatives of the observations can be also added to the original model for better predictions and to improve the computational efficiency of the model.

This chapter is a summary of the concept of Gaussian Processes with Derivative Observations and is based on the paper Solak and Leith (2002).

Let the underlying system is described as

$$\boldsymbol{y} = f(\boldsymbol{x}) + v \tag{2.15}$$

and the function f is modeled by a Gaussian process as

$$f \sim \mathcal{GP}(m(\boldsymbol{x}), K(\boldsymbol{x},\boldsymbol{x}')) \tag{2.16}$$

For simplicity, the mean function is assumed $m(\boldsymbol{x}) = 0$ and the Squared Exponential Kernel is used for the covariance matrix with some Gaussian noise v in the observations as

$$cov[y,y'] = K(\boldsymbol{x},\boldsymbol{x}') = \alpha\exp\left(-\frac{1}{2}\|\boldsymbol{x}-\boldsymbol{x}'\|_\Gamma^2\right) + v\delta_{m,n} \tag{2.17}$$

where $\delta_{m,n}$ is the Kronecker delta

$$\delta_{m,n} = \begin{cases} 1 & m = n \\ 0 & m \neq n \end{cases} \tag{2.18}$$

and the norm $\|\cdot\|_\Gamma$ defined as

$$\|\boldsymbol{x}\|_\Gamma = (x'\Gamma x)^{\frac{1}{2}}, \Gamma = \begin{bmatrix} \gamma_1 & & \\ & \ddots & \\ & & \gamma_D \end{bmatrix} \tag{2.19}$$

is the Euclidean distance with some scaling factor γ_i along each dimension of the vector $\boldsymbol{x}$.

Let the partial derivatives for the initial M observations $\boldsymbol{X} = [\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^M], \boldsymbol{x} \in R^D$ are given as $\Omega_i = [\omega^{i,1}, \omega^{i,2}, \ldots, \omega^{i,M}]$ for $i = 1, \ldots, D$ where the index i identifies the $i-$th partial derivative with respect to the $i-$th dimension of the initial observations $\boldsymbol{X}$. In other words, the $i-$th partial derivative with respect to the $m-$th observation is defined as

$$\omega^{i,m} = \frac{\partial f(\boldsymbol{x})}{\partial x_i}|_{\boldsymbol{x}=x^{i,m}}, i = 1, \ldots, D, m = 1, \ldots, M \tag{2.20}$$

Note that for the $\omega^{i,m}$ only the $i-$th component of the $m-$th observation $x^{i,m}$ is non zero.

To find the joint probability of the true observed values $\boldsymbol{y} = [y^1, y^2, \ldots, y^M]$ and the partial derivatives $\Omega_i = [\omega^{i,1}, \omega^{i,2}, \ldots, \omega^{i,M}], i = 1, \ldots, D$ (both correponding to the M initial observations $\boldsymbol{X} = [\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^M]$) it is needed to calculate the covariance between the function and the derivative observations. The covariance matrix is a linear mapping function so it is generally differentiable (some of the Matern Kernel functions are not). The following relations for the covatiance matrix are true

$$cov[\omega^{i,m}, y^n] = \frac{\partial}{\partial x_i} cov[y^m, y^n] \tag{2.21}$$

$$cov[\omega^{i,m}, \omega^{j,n}] = \frac{\partial^2}{\partial x_i \partial x_j} cov[y^m, y^n] \tag{2.22}$$

For the entries of the Squared Exponent Kernel (in the noise-free case) these relations are given as

$$cov[y^m, y^n] = \alpha \exp\left(-\frac{1}{2}\|\boldsymbol{x}^m - \boldsymbol{x}^n\|_\Gamma^2\right) = K_{xx} \tag{2.23}$$

$$cov[\omega^{i,m}, y^n] = -\alpha\gamma_i(x_i^m - x_i^n) cov[y^m, y^n] = K_{dx} \tag{2.24}$$

$$cov[\omega^{i,m}, \omega^{j,n}] = \alpha\gamma_i(\delta_{i,j} - \gamma_j(x_i^m - x_i^n)(x_j^m - x_j^n)) cov[y^m, y^n] = K_{dd} \tag{2.25}$$

Arranging the initial observations and their output values and partial derivatives as

$$\boldsymbol{X} = [\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^N], \boldsymbol{x} \in R^D$$
$$\boldsymbol{y} = [y^1, y^2, \ldots, y^N, \omega^{j,1}, \omega^{j,2}, \ldots, \omega^{j,N}], j = 1, \ldots, D$$

results in new covariance matrix which is now split into 4 parts as

$$K^*(\boldsymbol{X}, \boldsymbol{X}') = \begin{bmatrix} \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & K_{xx}(\boldsymbol{x}^m, \boldsymbol{x}^n) & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix}_{\substack{m\in\{1,\ldots,M\} \\ n\in\{1,\ldots,N\}}} & \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & K_{dx}(\boldsymbol{x}_i^m, \boldsymbol{x}^n) & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix}_{\substack{m\in\{1,\ldots,M\} \\ n\in\{1,\ldots,N\} \\ i\in\{1,\ldots,D\}}} \\ \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & K_{xd}(\boldsymbol{x}^m, \boldsymbol{x}_j^n) & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix}_{\substack{m\in\{1,\ldots,M\} \\ n\in\{1,\ldots,N\} \\ j\in\{1,\ldots,D\}}} & \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & K_{dd}(\boldsymbol{x}_i^m, \boldsymbol{x}_j^n) & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix}_{\substack{m\in\{1,\ldots,M\} \\ n\in\{1,\ldots,N\} \\ i\in\{1,\ldots,D\} \\ j\in\{1,\ldots,D\}}} \end{bmatrix} \tag{2.26}$$

where the size of matrix $K_{xx}(\boldsymbol{x}^m, \boldsymbol{x}^n)$ is $M \times N$, $K_{dx}(\boldsymbol{x}_j^m, \boldsymbol{x}^n)$ is $M \times ND$, $K_{xd}(\boldsymbol{x}^m, \boldsymbol{x}_i^n)$ is $MD \times N$, $K_{dd}(\boldsymbol{x}_j^m, \boldsymbol{x}_i^n)$ is $MD \times ND$ and $K^*(\boldsymbol{X}, \boldsymbol{X}')$ is $(MD + M) \times (ND + N)$ and again i and j denote partial derivatives with respect to the the i-th and j-th components of the input space.

From subsection 2.4.2 the Gaussian process predictions are still described as

$$f_* \sim \mathcal{N}(\boldsymbol{\mu}_*, \Sigma_*) \tag{2.27}$$

with the new covariance function K^* from Equation 2.26 the mean is now

$$\boldsymbol{\mu}_* = K^*(\boldsymbol{X}_*, \boldsymbol{X}) K^*(\boldsymbol{X}, \boldsymbol{X})^{-1} \boldsymbol{y} \tag{2.28}$$

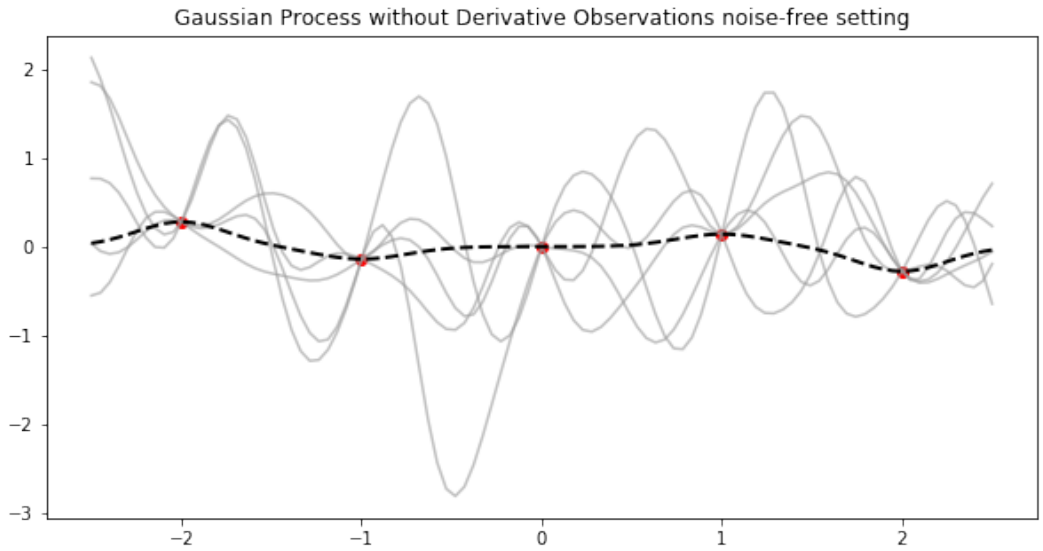
and the covariance matrix

$$\Sigma_* = K(\boldsymbol{X}_*, \boldsymbol{X}_*) - K^*(\boldsymbol{X}_*, \boldsymbol{X}) K^*(\boldsymbol{X}, \boldsymbol{X})^{-1} K^*(\boldsymbol{X}, \boldsymbol{X}_*) \tag{2.29}$$

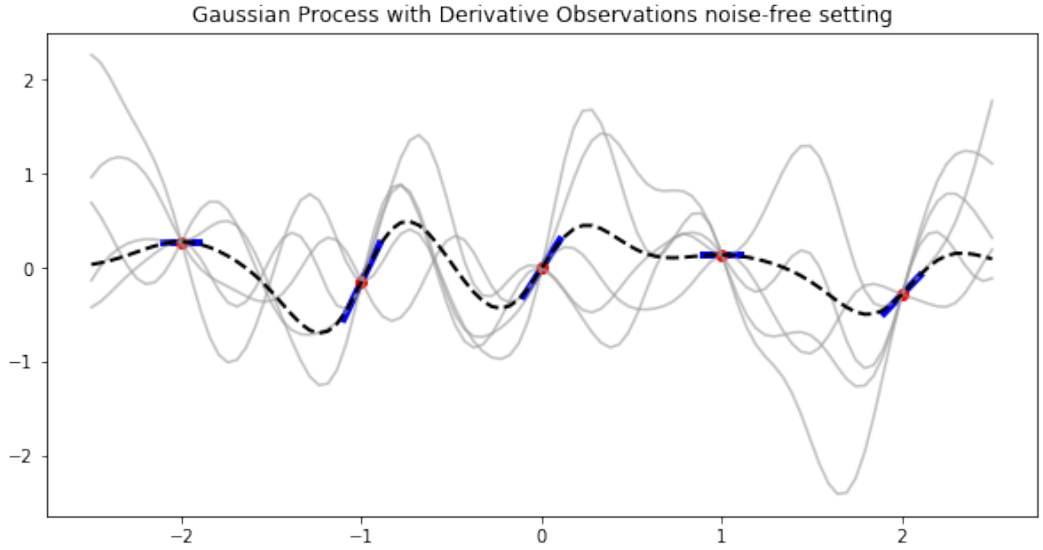
where $\boldsymbol{X}_*$ is the set of unknown datapoints (test), $\boldsymbol{X}$ is the set of initial observations and

$$K^*(\boldsymbol{X}_*, \boldsymbol{X}) = \left[\begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & K_{xx}(\boldsymbol{x}_*^m, \boldsymbol{x}^n) & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix}_{\substack{m \in \{1,\ldots,M\} \\ n \in \{1,\ldots,N\}}} \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & K_{xd}(\boldsymbol{x}_*^m, \boldsymbol{x}_j^n) & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix}_{\substack{m \in \{1,\ldots,M\} \\ n \in \{1,\ldots,N\} \\ j \in \{1,\ldots,D\}}} \right] \tag{2.30}$$

The effect of applying derivative observation to the system are graphically shown in Figure 2.10



(a) *Gaussian process on a system without derivative observations.*



(b) *Gaussian process with derivative observations. The model is additionally constrained at the known points by their derivatives.*

Figure 2.10: *The effect of applying derivative observations at the observation points.*

3 Analysis

Single cell RNA sequencing technology is used in a broad range of scientific research questions including gene expression profiling, studying of splicing events associated with diseases and most recent applications emerging in the study of complex biological processes, cellular heterogeneity, and diversity in cells. It has introduced a lot of possibilities and even more questions regarding the complexity of biological systems and their development. In this chapter the scope of the problem is defined, the proposed solution is explained and various alternatives and limitations are discussed.

3.1 Problem Domain

In this dissertation the focus will be on the use of scRNA-seq in combination with Gaussian processes and RNA Velocity for the profiling and unsupervised ranking of genes based on their smoothness. Smooth genes are identified as the genes whose gene expression is varying smoothly across all the cells. The identification of such genes is of particular interest as this is a form of gene profiling and can unravel interesting hidden structures in the dataset as well as biological properties of the system. For example, the approach presented in this dissertation discovers in an unsupervized manner a set of smoothly varying genes resulting from a TGFB1-induced differentiation event in human vein cells as described by EL Low (2019) (see section 6.1).

3.1.1 Identifying Smooth Genes with Gaussian Process

The decision to use Gaussian process for this task is based on the properties of the model. As described in section 2.4 the Gaussian process is absolutely defined by its covariance matrix and mean function. Almost all of the known covariance functions are dependent on the distances between the input space variables resulting in a distance correlation between the variables. This means the Gaussian process can model both linear and non-linear dependence and for almost all kernel functions the correlation decays smoothly as a function of distance.

Most of the covariance functions (kernels) used for the Gaussian process are differentiable. As smoothness is the main target of the model, for this particular problem choosing the Squared Exponential Kernel (aka the RBF kernel) as defined in Equation 2.12 is feasible since this kernel is infinitely differentiable. In the calculus sense this means the latent space defined by the Gaussian process has derivatives of all orders making it very smooth.

The level of desired smoothness is dependent on the chosen kernel function, its hyperparameters and the subjective objectives of the problem: sometimes smooth results are preferred and sometimes less smooth results indicate a more realistic product. Thus, other families of kernels such as the Matern Kernel which provide a finer control over the differentiability can also be of interest.

The scRNA-seq data is highly dimensional so a mapping in lower dimensions is needed. For this type of data common dimensionality reduction algorithms are used and described in section 2.2. The input space is now in 2D (the cells coordinates) and the output space is the gene intensity (a scalar) resulting in a set of different surfaces for each gene. Gaussian processes are particularly

good at modeling surfaces and working with this model for predictions in this subspace will still carry important information for the genes and their smoothness.

Furthermore, the scRNA-seq data suffers from both technical and biological noise which imposes further challenges when identifying the smooth genes. This is especially important when the intensity of the gene expressions are low. A great property of the Gaussian process is that it can incorporate the noise of the observations by simply specifying the noise variable in the kernel function without changing the other properties of the kernel function.

For all the aforementioned reasons, Gaussian processes are a great candidate for basis of an efficient solution to finding smooth genes. Due to the smoothness of the covariance matrix, the ability to deal with the very noisy dataset and the variety of kernel families, Gaussian processes are particularly good for identifying smooth genes where the definition of 'smooth' can be adjusted to the specific problem objective.

3.1.2 Unsupervized Ranking of Genes

As described above, using Gaussian processes is a reasonable approach for finding smooth genes but having a way to rank the genes based on their smoothness is even more helpful for analysing purposes. Aiming for an unsupervized way of identifying and ranking the genes of interest (the smooth ones) can result in a family of genes which may be of biological importance to the system, closely correlated in a way, both or neither.

The Gaussian process have a well-defined log marginal likelihood which among other properties is a meaningful criteria for how good a fit the data is to the model. Higher likelihood for the given Gaussian process, kernel function and hyperparameters means better fit of the data therefore smoother gene. Thus, computing the likelihood of the gene and using it as a ranking criteria is sufficient for unsupervized ranking of the genes.

3.1.3 Aiding Gaussian Process with RNA Velocity

RNA Velocity as defined by La Manno (2018); Volker Bergen (2019) is the time derivative of the gene expression state and is obtained from the deviation of the observed ratio of spliced and unspliced mRNA molecules. The RNA Velocity introduces additional information for the state of the system and predicts the further differentiation of the cells. RNA Velocity can capture further dependency of the genes within the cell and the genes over all cells because cell's fate determination is regulated by the genes. The global differentiation of the cells into clusters adds further information for the genes and their future variance across the cells. This feature can be particularly useful for low intensity genes.

There are three main RNA Velocity models: steady-state model, stochastic model and dynamic model. The latter one allows comparison of velocities on a gene level. Nevertheless, all three models capture information for the global differentiation of the cells thus introducing additional information for the genes.

Gaussian processes incorporate derivative observations and RNA Velocity can be substituted for a derivative observation as it has both a direction and a magnitude that carry meaning for the future movement of the system. RNA Velocity shows the direction in which cells are differentiating therefore it will constrain the Gaussian process and favour the genes which are varying smoothly (gradually developing) towards the cell fate determination state.

Thus, integrating RNA velocities as another information to the original model can result in:

1. Reduction of the number of observations needed as it introduces additional information;
2. Speed up of the computations;
3. Better fit of the data to the model (smoother);

4. Identification of low intensity smooth genes which have been potentially ignored by the Baseline Gaussian Process model;
5. Boost of genes which are gradually developing in the direction of the cells fate determination state (RNA Velocity).

3.2 Alternatives to Gaussian Process

There are other models that introduce this idea of smoothness like the Ordinary Kriging Regression model. This is an interpolation method which also takes into account that the mean of the posterior distribution might not be zero. In comparison with the Gaussian process, the Ordinary Kriging Regression extrapolates better. However, for this particular problem the output range is constrained by the cells in the dataset. Thus, better extrapolation will not significantly benefit the model as there is no way of verifying those predictions. Therefore, the Ordinary Kriging Regression is not expected to have any significant advantage over Gaussian process.

3.3 Problems Regarding Gaussian Process

With the increasing popularity of scRNA-seq and the introduction of new protocols there is a tendency the size of the scRNA-seq datasets to increase. This presents computational challenge for complex models like the Gaussian process. A big weakness of the Gaussian process is that the computations needed for the model include the inversion of the covariance matrix. The process is of (naively) time complexity $O(n^3)$ which becomes a computational burden and practically unusable when the number of initial observations is too big.

This can be potentially avoided by using the Gaussian Process with RNA Velocity observations and much less amount of initial observations. Another approach is using only a portion of the initial observations that comprise good representation of the whole dataset. For both approaches the prior covariance matrix and its inverse are cached as the input space (the low dimensional mapping of the cells) does not change for the genes.

Finally, the Gaussian Process model assumes Gaussian likelihood of the initial observations. However, this is very often not the case with scRNA-seq datasets due to the high volume of drop-outs and the non negative counts. The system incorporating such information results in an intractable posterior distribution due to non-conjugate Gaussian process prior to non-Gaussian likelihood. However, the benefit of the implementation presented in this dissertation is that the marginal likelihood and the posterior distribution are analytically computed resulting in fast and exact computation.

4 Design

4.1 Basic Normalization and Filtering

The firs step of the design process involves obtaining the raw count matrix from the text files containing the scRNA-seq sequences with the help of specialized string alignment software. Once the raw count matrix is extracted it is cleaned up following the pipeline steps described in Figure 2.2 and with the suitable libraries for that the data is transformed as follows:

1. Filter out outliers some of which are defined as:
   (a) Cells with low number of genes;
   (b) Genes found only in low number of cells;
   (c) Cells with high number of gene counts;
   (d) Cells with high number of mitochondrial genes;
2. Normalize the counts by the total number of counts so that genes can be comparable across cells;
3. Logarithmize the count matrix so that the variance on the first principal component is reduced and the distribution of the expression values more normal;
4. Conduct further filtering based on the highly variable genes if needed.

Once the data has been cleaned it is saved externally so that it can be used multiple times for different analysis without the need to compute the basic filtering and normalization every time.

4.2 Dimensionality Reduction and Clustering

Following that it is proceeded to dimensionality reduction techniques in order to find hidden structures in the dataset. The computations are obtained with the help of the PCA, t-SNE and/or UMAP algorithms. The results from the dimensionality reduction are plotted and the cells are assigned to clusters using graph-clustering methods. Another checkpoint is created and the results from the aforementioned methods are saved externally.

4.3 RNA Velocity

Once the data is presented in a meaningful manner velocity analysis is executed. Depending on the dataset and the objectives of the experiment different velocity computations are obtained (steady-state model, stochastic model or dynamic model). As velocity analysis is time consuming and computationally heavy another checkpoint is created.

4.4 Gaussian Process Models

Having obtained the velocities and the low dimensional mapping the data is run through the Gaussian Process models. This step presents the unsupervised learning of the rankings of the gene expressions based on their smoothness. The smoothness is defined as the quality of the model's

fit to the dataset for each gene and it's numerically calculated as the log marginal likelihood. The bigger likelihood corresponds to better fit of the model and smoother gene. The input of the Gaussian process model is the mapping of the cells in 2D space obtained via dimensionality reduction algorithms and the predictions are the gene expression in each cell for each gene. There are two Gaussian Process models implemented for this dataset:

1. Baseline Gaussian Process;
2. Baseline Gaussian Process with Derivative Observations or adding RNA Velocity as additional information to the original model.

Only a subset of the cells are fed to the model for the training step for faster computations. Choosing between 20% to 50% of the cells proves to be efficient and good enough presentation of the whole dataset. For the Baseline Gaussian Process with Derivative Observations model even smaller portion of the cells (between 20% and 30% of the whole dataset) is sufficient.

4.5 Result Analysis

The results from the models are examined. A report of the smoothest and least smooth genes is created and the genes are inspected for biological meaning. In addition, other properties for the genes of interest (top 10 and last 10 by smoothness) are investigated. These properties include:

1. The behaviour of the gene in specific clusters and across all cells;
2. The phase portrait of the specific gene;
3. Correlation with other genes.

Comparison between the rankings of the two models is also conducted in order to search for genes which have been significantly favoured by the RNA velocity model and their properties.

Finally, to evaluate the models the root of the mean squared error (MSE) is used as a criteria to check which model performs better and is a better fit in general. The results are evaluated using the paired t-test.

The whole process is summarized in the flowchart shown in Figure 4.1.

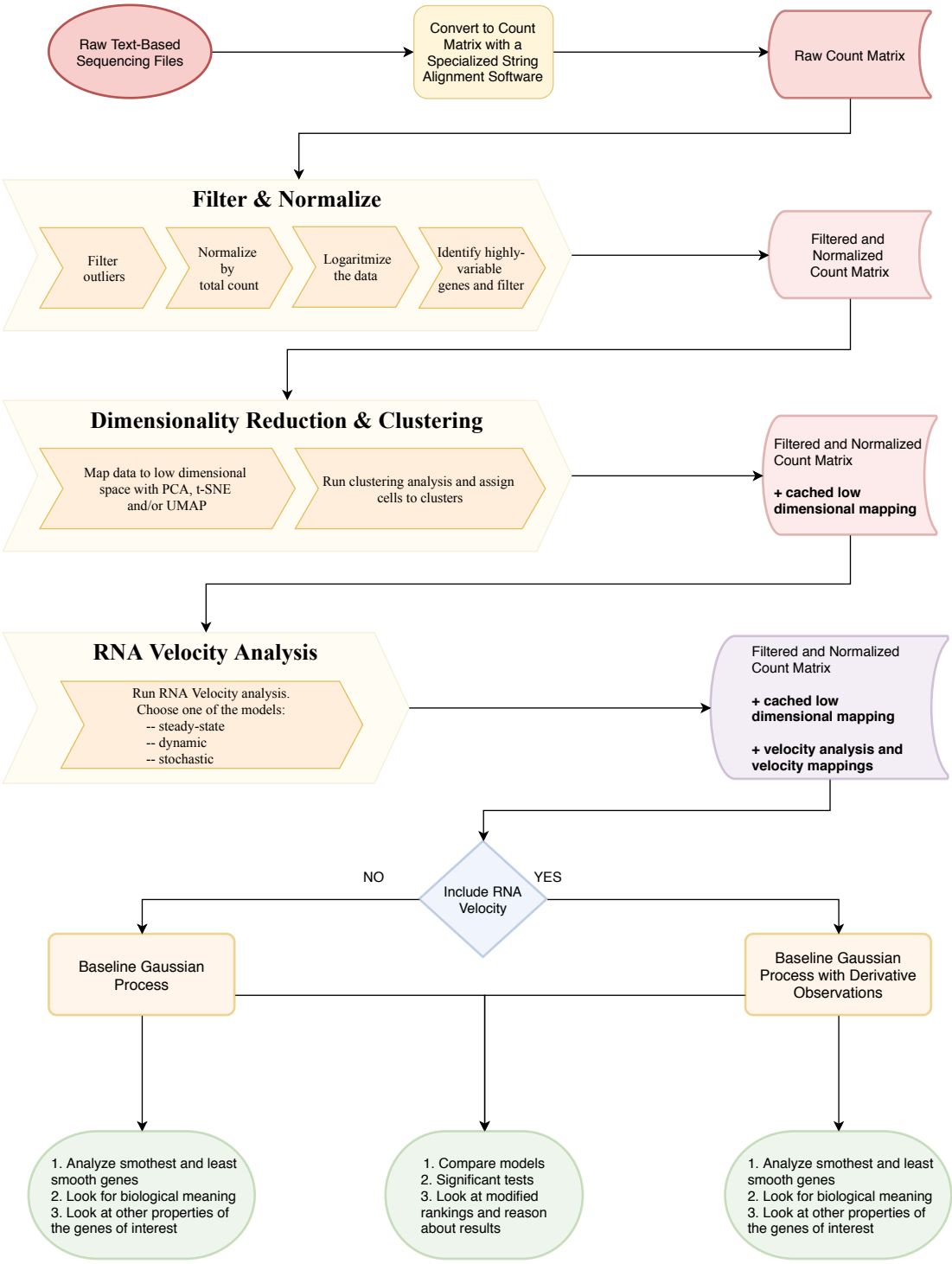


Figure 4.1: *A summarized overview of the design and the processing and analysis of the datasets.*

5 Implementation

This chapter explains the implementation of the processing pipeline for scRNA-seq datasets as explained in Figure 2.2. Python was chosen as a language for the implementations because of its availability and convenience for big data processing. In addition, many of the tools used for computing RNA Velocity and scRNA-seq data processing have Python APIs. All the experiments are conducted in Jupyter Notebooks (Python environment), which enable convenient visualisation of the graphs and can be easily uploaded to Colab or Binder and be run and shared online.

5.1 Data Preprocessing

For the preprocessing of the datasets two main libraries are used: scanpy and scvelo. Scanpy originally introduced by Wolf (2018) is a scalable toolkit specifically developed for single cell RNA sequencing datasets. It provides all the necessary functionality for processing such as

1. Filtering outlier cells based on total counts within the cell or the numbers of expressed genes within the cell;
2. Filtering outlier genes by the number of cells they have been expressed in or the total counts of the gene
3. Identifications of highly variable genes;
4. Normalization of the count matrix and others.

In addition, the library supports the three main low dimensionality reduction embedding algorithms: PCA, t-SNE and UMAP. There are also visualization methods such as violin plots and embedding results that are very useful when checking each step of the preprocessing.

RNA Velocity analysis are obtained with scvelo. This library introduced by Volker Bergen (2019) is compatible with scanpy which makes it very convenient for the computations of RNA Velocities. It supports three different models: 'steady-state', 'stochastic' and 'dynamical' (see subsection 2.3.2 and subsection 2.3.3). For this dissertation, mainly the 'stochastic' and the 'dynamic' models have been used. The 'stochastic' model captures to a greater extend the full dynamics of system and at the same time is significantly faster than the 'dynamical' model. Thus, the 'stochastic' model is used when the datasets are quite big with lots of genes.

5.2 Kernels

There are a couple of abstract classes implemented for the kernels.

1. The abstract `Kernel` class;
2. Wrapper `Sum` and `Prod` classes.

The abstract `Kernel` class is used as a base for the implementation of all other kernels. It provides methods for the summation and product of different kernels. In addition, it ensures that these operations are done on strictly `Kernel` objects. The wrapper classes `Prod` and `Sum` are used for the actual implementation of the summation and product between `Kernel` objects.

There are three main kernels implemented as separate classes: `RBFKernel`, `Matern52Kernel` and `LinearKernel`. The `RBFKernel` is essentially the Squared Exponential kernel as defined in Equation 2.12. The `Matern52Kernel` is evaluated at $\nu = \frac{5}{2}$ and is defined as

$$K(\boldsymbol{X}, \boldsymbol{X}') = \alpha(1 + \sqrt{5}\|\boldsymbol{x} - \boldsymbol{x}'\|_\Gamma + \frac{5}{3}\|\boldsymbol{x} - \boldsymbol{x}'\|_\Gamma^2) \exp(-\sqrt{5}\|\boldsymbol{x} - \boldsymbol{x}'\|_\Gamma) \tag{5.1}$$

where α is the scaling factor and $\|\boldsymbol{x} - \boldsymbol{x}'\|_\Gamma$ is Euclidean distance between the vector $\boldsymbol{x}$ and $\boldsymbol{x}'$ scaled by some constant constants $\boldsymbol{\Gamma} = [\gamma_1, ..., \gamma_D]$ for each dimension of $\boldsymbol{x}$ and $\boldsymbol{x}'$ respectively. Only the `Matern52Kernel` is implemented from the kernels defined in the Matern family because it is double differential and allows derivative observations.

The `LinearKernel` is simply defined as the dot product between the input points as

$$K(\boldsymbol{X}, \boldsymbol{X}') = \sigma^2 + \alpha \boldsymbol{x}^T \boldsymbol{x}' \tag{5.2}$$

where σ is the parameter controlling the homogeneity of the kernel and α is the scaling factor.

In addition, the derivative with respect to the hyperparameters defining each kernel is provided as a separate method. This is used in the optimization of the hyperparaemters.

For the implementation of the kernels the `numpy` package is used as the code is highly vecotrizable. In addition, helper packages of the scientific library `scipy` introduced by Virtanen et al. (2020) are used for the fast computation of distances between input points.

5.3 Baseline Gaussian Process

The baseline `GaussianProcess` class is a general implementation of the Gaussian Process Regression model. There are some customizations of the class that speed up the computations specifically for scRNA-seq datasets. The class is compatible with external kernel classes as defined by `sklearn` and with the kernel classes described above. The class methods of the `GaussianProcess` class comprise the normal functionality of Gaussian Process Regression:

1. A `fit` method for fitting the data;
2. A `predict` method for predictions on given data;
3. A `sample` method for sampling from the posterior distribution on the given data.

Some of the customizations for scRNA-seq data include:

1. The class allows the user to specify the amount of data to be used for the fitting with the `sample_ratio` parameter (0.0 `sample_ratio` ≤ 1.0);
2. The covariance matrix and its inverse are cached for each model as input cells are always the same (which results in huge speed-up for the computations, about 150 times faster than other libraries);
3. A `log_marginal_likelihood` method for computation of the log marginal likelihood for the given gene. This method assumes the most recent fit of the dataset is for the given gene;
4. A `get_all_likelihoods` method that returns a dictionary of the genes and their likelihoods.

5.4 Gaussian Process with RNA Velocity Observations

To accommodate RNA Velocity Observations (derivative observations), a few modifications to the original class `GaussianProcess` and the `RBFKernel`, `Matern52Kernel` and `LinearKernel` classes were adopted. For the `GaussianProcess` these are

1. An additional boolean attribute `derivative_observations` which specifies whether to include derivative observations or not;
2. Modifications along the code to pass the new attribute `derivative_observations` when the kernel function is invoked;
3. The `sample_ratio` attribute is also applied to the derivative observations of the input data.

For the kernel classes the modifications were bigger and are as specified below.

1. Additional computation for the partial derivatives with respect to each dimension of the input data.
2. A new composite matrix consisting of the original kernel matrix and the new parts including derivative observations (see Equation 2.26) when the `derivative_observations` is set to `True` and the kernel is invoked on the input space.
3. A new composite matrix of the form shown in Equation 2.30 when the `derivative_observations` is set to `True` and the kernel is invoked on the test and train data.

To include derivative observations, the new covariance matrix is constructed as in Equation 2.26. The algorithm for constructing the right matrix is shown below.

Data: $K(X, X')$, a kernel function; X, X', matrices of vector observations (could be train and train, test and train, and test and test);
Result: Cov, a composite covariance matrix with derivative observations.
begin
  $Cov_{wy} \longleftarrow []$
  $Cov_{ww} \longleftarrow []$
  $D \longleftarrow X.dim$
  for $i < D$ **do**
    $Cov_{w_i y} \longleftarrow$ the partial derivative of the kernel w.r.t the i-th dimension
    append $Cov_{w_i y}$ to Cov_{wy}
  end
  for $i < D$ **do**
    for $j < D$ **do**
      $Cov_{w_i w_j} \longleftarrow$ the partial derivative of the kernel w.r.t the i-th and j-th dimension
      append $Cov_{w_i w_j}$ to Cov_{ww}
    end
  end
  if X *and* X' *are test data* **then**
    $Cov \longleftarrow \begin{bmatrix} Cov_{yy} \end{bmatrix}$
  end

  if X *is test data and* X' *is train data* **then**
    $Cov \longleftarrow \begin{bmatrix} Cov_{yy} & Cov_{yw} \end{bmatrix}$
  end

  if X *and* X' *are train data* **then**
    $Cov \longleftarrow \begin{bmatrix} Cov_{yy} & Cov_{wy} \\ Cov_{yw} & Cov_{ww} \end{bmatrix}$
  end

  return Cov
end

The gradient of the kernel function w.r.t the hyperparameters is computed only for the non-derivative part $\begin{bmatrix} Cov_{yy} \end{bmatrix}$ in both cases (with and without derivative observations). This is a trade-off between gradient correctness and computational time for the optimization of the hyperparameters.

6 Evaluation

This chapter discusses the final results obtained from the Gaussian Process models and evaluates their performance against different scRNA-seq datasets. The three tested models are Baseline Gaussian Process, Gaussian Process with Composite Kernel, and Gaussian Process with RNA Velocity Observations. Two datasets are used for the evaluation: the Human Saphenous Vein dataset based on the 10x Genomics protocol (section 6.1) and the Dentate Gyrus dataset also based on the 10x Genomics protocol (section 6.2). In addition, the properties of the genes identified by the models are discussed. A brief summary of the Gaussian process models and a comparison of their properties is shown in Table 6.7.

6.1 Human Saphenous Vein Dataset

This scRNA-seq dataset comprises human saphenous vein smooth muscle cells from EL Low (2019). The isolated cells are split into two batches: the first batch is left as it is; the second batch is treated with protein 'TGFB1'. The protein induces a differential event in the cells' maturation and for this reason analysis are shown only for the treated batch (see A for the normal batch). The dataset is obtained with the 10x Genomics Protocol.

6.1.1 Dataset Analysis and Hyperparameters Choice

After preprocessing the dataset consists of 7441 cells with 4543 genes. Dataset is quite dense so using only a subset of the cells is sufficient. Picking 40% of all cells at random proves to be both efficient and a good presentation of the whole dataset. This speeds up the computations 5 times and gives identical results.

For the baseline Gaussian process model the Squared Exponential (RBF) kernel is used due to its smoothness properties (see subsection 3.1.1). Initial parameters of the Squared Exponential kernel are set as $\alpha = 0.1$ (as data is logaritmized thus scaling is applied) and $\gamma = 1.0$. Random gene is selected for the optimization of the γ parameter and the optimized $\gamma = 1.0$ (which has not changed in the optimization part) is used for the rest of the genes.

The second model Gaussian process with composite kernel uses the sum of the Squared Exponential Kernel and the Linear Kernel as its covariance matrix. The introduced linearity forces the model to interpolate between the observation points more gradually (linearly) thus making the fit smoother where gaps in the reads are present. The model is using the optimized $\gamma = 1.0$ and $\alpha = 0.1$ from the baseline model for the RBF kernel. For the linear part of the composite kernel parameters are set to $sigma = 0.0$ as the offset and $scale = 0.1$ so that it is comparable to the α parameter of the RBF kernel. The model is still fit to 40% of the dataset.

The final model Gaussian Process with RNA Velocity Observations uses the optimized baseline model hyperparameters $\alpha = 0.1$ and $\gamma = 1.0$. The RNA Velocity is used as derivative observations for each cell. By doing so the RNA Veocity is constraining the model at the observation points and it is interpolating between the observation points in a similar manner as is the second model. However, the slope at each observation point (cell) might be completely different than the slope described by 2 neighbouring cells resulting in very different results (the effect of derivative

observations is shown in Figure 2.10). The RNA Velocity is bringing new information so less data is used (only 20% of the original dataset).

The noise parameter in all three models is set to $\alpha_{noise} = 1e - 10$ as the data is quite dense and drop-outs have been removed in the preprocessing part (there are no 0 values). In addition, this adds numerical stability when computing the Cholesky decomposition of the covariance matrix.

The optimization of the hyperparameters is very costly as the covariance matrix and it's inverse are computed at each step of the optimization part. For this reason, all three models use the optimizationed hyperparameters from the baseline Gaussian process model level. This approach is considered reasonable, because the dataset does not change and thus neither do its properties. Using the same smoothness parameters for the Squared Exponential Kernel across models is therefore not detrimental.

6.1.2 Results and Analysis

The low dimensional mapping used for this dataset is UMAP as it is the state of the art dimensionality reduction algorithm for scRNA-seq datasets. The top ten smoothest and least smooth genes after running the Baseline Gaussian process model are shown in Table 6.1

Baseline Gaussian Process Model			
Smoothest Genes		**Least Smooth Genes**	
1.*CALD*1	2.*TNFRSF*11*B*1	1.*KLHL*4	2.*LDLRAD*4
3.*SERPINE*1	4.*CCDC*80	3.*RAB*27*B*	4.*LINC*01139
5.*PXDC*1	6.*DDAH*1	5.*DNM*1	6.*SEPT*6
7.*WWTR*1	8.*EXT*1	7.*KRT*8	8.*DEPTOR*
9.*FGF*2	10.*DLC*1	9.*AC*083967.1	10.*RTKN*2

Table 6.1: *Top 10 smoothest and least smooth genes after running the Baseline Gaussian Model on 40% of the Human Saphenous Vein dataset.*

Some important genes identified by the model as very smooth, such as 'SERPINE1', have been extracted manually or using t-test and other statistical methods. However, the baseline Gaussian process model is able to identify this gene along with others in an unsupervized way.

The data is also fit to the Gaussian Process with Composite Kernel (Linear + Squared Exponent) and Gaussian Process with RNA Velocity observations. The smoothest genes ranked by the models are shown respectively in Table 6.2 and Table 6.3.

Gaussian Process Model with Composite Kernel			
Smoothest Genes		**Least Smooth Genes**	
1.*CALD*1	2.*SERPINE*1	1.*LINC*01139	2.*ERVMER*61 − 1
3.*CCDC*80	4.*TNFRSF*11*B*1	3.*HAPLN*1	4.*CPM*
5.*PXDC*1	6.*EXT*1	5.*LRRC*2	6.*BMF*
7.*DDAH*1	8.*DLC*1	7.*AC*003092.1	8.*CDCP*1
9.*FST*	10.*ITGBL*1	9.*ANKRD*45	10.*AKR*1*B*10

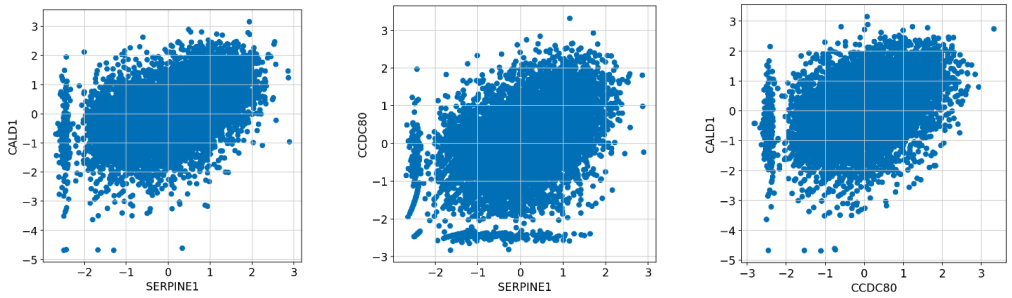
Table 6.2: *Top 10 smoothest and least smooth genes after running the Gaussian Model with Composite Kernel on 40% of the Human Saphenous Vein dataset.*

As seen from Table 6.1, Table 6.2, Table 6.3 there is some re-ranking of the genes compared to the baseline model, however, all three models agree on most of them.

Gaussian Process Model with RNA Velocity Observations			
Smoothest Genes		**Least Smooth Genes**	
1.*CALD*1	2.*SERPINE*1	1.*SEMA3D*	2.*AL*355607.2
3.*DDAH*1	4.*TNFRSF*11*B*	3.*VIPR*1	4.*LINC*01239
5.*CCDC*80	6.*FGF*2	5.*HAPLN*1	6.*DRAXIN*
7.*PXDC*1	8.*FST*	7.*PKIB*	8.*EDN*1
9.*DLC*1	10.*EXT*1	9.*PRG*4	10.*SNTB*1

Table 6.3: *Top 10 smoothest and least smooth genes after running the Gaussian Model with RNA Velocity Observations on 20% of the Human Saphenous Vein dataset.*

It is expected that smooth genes have high positive correlation and less smooth genes have low correlation. The correlation between three of the smoothest genes similarly identified by all three models is shown in Figure 6.1



(a) *Correlation between gene SERPINE1 and CALD1.*

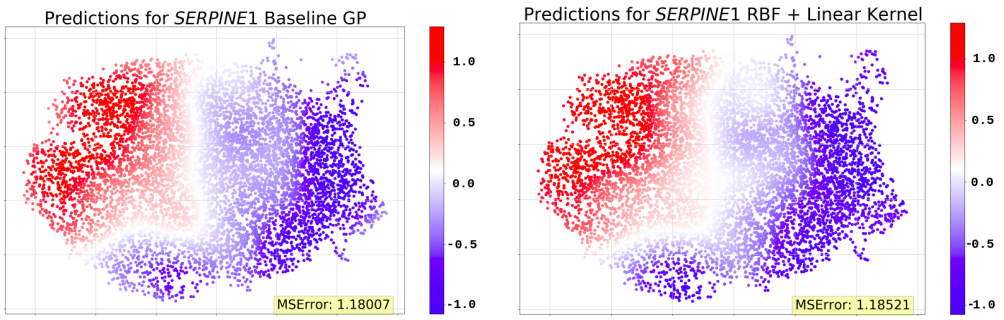
(b) *Correlation between gene SERPINE1 and CCDC80.*

(c) *Correlation between gene CCDC80 and CALD1.*

Figure 6.1: *The correlation from the top 3 smoothest genes as discovered by the baseline Gaussian process model. There is a clear correlation between the smoothest genes.*

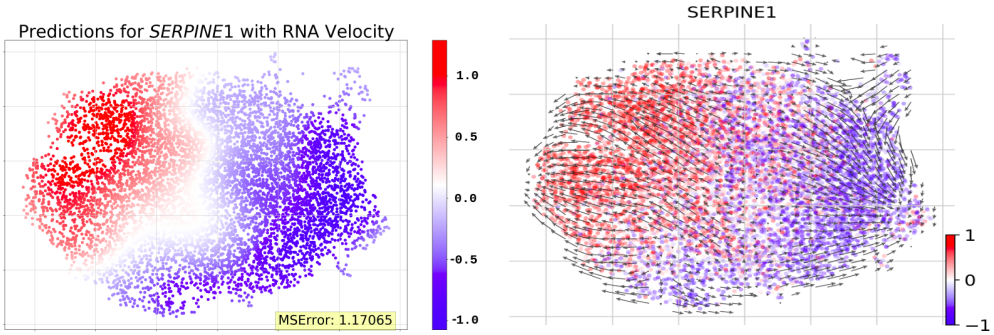
Training is only on a subset of the data so the gene expression functions can be tested on the whole dataset. Example expression function as derived by all three models for the 'SERPINE1' gene is shown in Figure 6.2

As seen from Figure 6.2 the best fit of the data with the smallest mean squared error (MSE) is obtained by the Gaussian Process with RNA Velocity observations. Both the Gaussian Process



(a) *The gene expression function predicted by the Baseline Gaussina Process.*

(b) *The gene expression function predicted by the GP with Composite Kernel.*



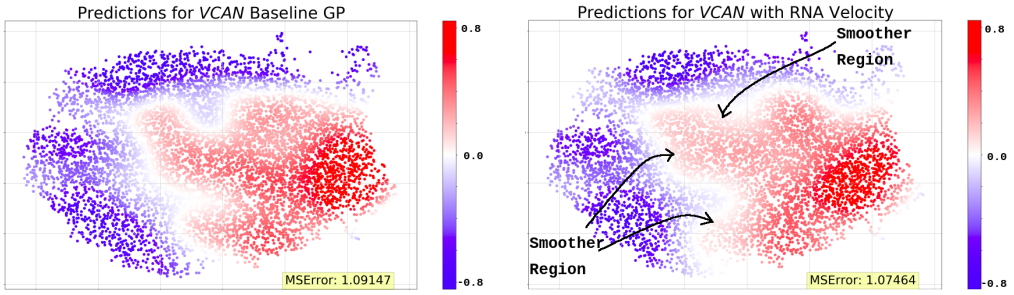
(c) The gene expression function predicted by the GP with RNA Velocity.

(d) The true values of the gene expression including the RNA velocity.

Figure 6.2: *The predicted gene expression function for the 'SERPINE1' gene. (d) shows the RNA Velocity UMA low dimensional map obtained by the 'stochastic' scvelo model.*

with RNA Velocity observations and the Gaussian Process with Composite Kernel model are smoothing the gene expression function, however, the Gaussian Process with RNA Velocity model is also taking into account the magnitude and the direction of the velocity. This results in a better fit which is consistent with the global cells' differentiation pathways.

When comparing the top fifty smoothest genes found by the Baseline Gaussian Process model to the gene rankings of Gaussian Process with RNA Velocity models some interesting properties have been found. Some of the top genes are ranked higher whereas others lower by the Gaussian Process with RNA Velocity model. Such genes are presented in Figure 6.3 and Figure 6.4 respectively.



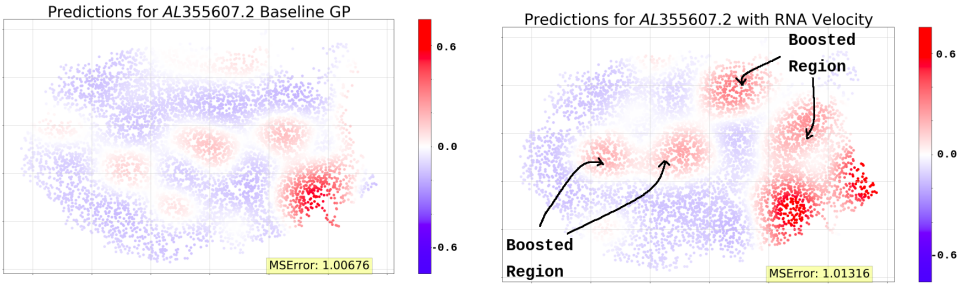
(a) The gene expression function for 'VGLL3' predicted by the Baseline Gaussina Process.

(b) The gene expression function for 'VGLL3' predicted by the GP with RNA Velocity.

Figure 6.3: *The predicted gene expression functions for the 'VGLL3' which has been ranked 12 places higher by the GP with RNA Velocity model. The model in (a) keeps the variations in the gene expression while the model in (b) smooths the gene at the shown regions due to velocity levels.*

As seen from Figure 6.3 some regions of the gene expression function have been smoothed by the velocity resulting in a better fit to the properties of the model. This results in some genes being ranked higher by the Gaussian Process with RNA Velocity model.

As seen from Figure 6.4 the velocity tends to boost the values at points with high velocity levels even when the reads of the gene are low. The velocity vector projection is global for all cells resulting in rapid smoothing of highly variable genes or boosting of gene expression values. Thus, an optimization for the scaling constant for the RNA velocity is needed.



(a) *The gene expression function for 'AL355607.2' predicted by the Baseline Gaussina Process.*

(b) *The gene expression function for 'AL355607.2' predicted by the GP with RNA Velocity.*

Figure 6.4: *The predicted gene expression functions for the 'AL355607.2' which has been ranked 400 places lower by the GP with RNA Velocity model. The model in (a) keeps the low gene expression values while the model in (b) boosts those values where velocity levels are high and lowers them where velocity levels are low.*

Running a paired two-tailed t-test at 95% confidence on the mean square errors from the Baseline Gaussian Process model vs the Gaussian Process model with RNA Velocity observations for each gene shows significant difference between the two models. The Gaussian Process model with RNA Velocity has smaller mean squared error (t-value is negative), thus being better. The value of t is -9.253104. The value of p is $< .00001$. The result is significant at $p < .05$.

6.2 Dentate Gyrus Dataset

The dataset consists of cells exracted from the Dentate Gyrus which is a part of the hippocampus associated with learning, memory formation and spacial coding. The dataset published by Hochgerner (2018) is obtained with the 10x Genomics protocol and consists of 25919 genes across 2930 cells.

6.2.1 Dataset Analysis and Hyperparameters Choice

Recreating the preprocesing pipeline as described in Volker Bergen (2019) results in 2894 genes across 2930 cells. Even after prerpocessing the dataset is extremely sparse with 78% of the reads being zeros (drop-outs). This is handled by the noise parameter ν of the Gaussian process. The dataset is fairly small thus 70% of the cells are used for the fitting of the models. This speeds up the computations twice and the results are identical.

For the Baseline Gaussian process with Squared Exponential (RBF) Kernel similar set-up is used as described in subsection 6.1.1. The optimized hyperparameters are $\gamma = 1.0$ and $\alpha = 0.1$. The noise parameter of the Gaussian process is set to $\alpha_{noise} = 1e - 4$ due to the high number of drop-outs.

The second model Gaussian process with composite kernel, being the sum of the Linear and the Squared Exponential kernel, is also using 70% of the dataset. The hyperparameters for the Squared Exponential kernel are set to the optimized $\gamma = 1.0$ and $\alpha = 0.1$ from the baseline Gaussian process model. The noise parameter is also set to $\alpha_{noise} = 1e - 4$. The Linear kernel parameters are set to $scale = 0.1$ and $sigma = 0.0$ similar to section 6.1.

The final model Gaussian Process with RNA Velocity observations is also fit on 70% of the cells because the dataset is very sparse. Since the total amount of cells is fairly low, the computations when including RNA Velocities is only 20% slower. The hyperparameters of the model are set as the optimized hyperparameters of the baseline model as discussed above.

6.2.2 Results

The default low dimensional mapping for this dataset is also UMAP and is used in this experiment. The top ten smoothest and least smooth genes identified by the baseline model are shown in Table 6.4.

Baseline Gaussian Process Model			
Smoothest Genes		**Least Smooth Genes**	
1.*Cntnap5c*	2.*Gm11508*	1.*Tmsb10*	2.*Slc25a4*
3.*Zfp931*	4.*Atp6ap1*	3.*Rps23*	4.*Rpl17a*
5.*Pm20d2*	6.*A430105J06Rik*	5.*Rpl37a*	6.*Rpl37*
7.*4930426I24Rik*	8.*Frmd3*	7.*Rps27*	8.*Rps16*
9.*Tgfbr2*	10.*Ccdc93*	9.*Rps24*	10.*Rps11*

Table 6.4: *Top 10 smoothest and least smooth genes after running the Baseline Model on the Dentate Gyrus dataset.*

The smoothest genes are present in only a very low number of cells (< 50 cells which is less than 1.4% of the cells). This shows a bias of the model to rank genes as smoothest when they have many zero counts. Nevertheless, almost all of the least smooth genes (expressed in ≥ 70% of the cells) found by the model are also found in the top velocity genes ranked by `scvelo`.

The data is also fit to the Gaussian Process with Composite Kernel (Linear + Squared Exponential) and the Gaussian Process with RNA Velocity Observations. The results are shown in Table A.1 and Table A.3 respectively.

Gaussian Process Model with Composite Kernel			
Smoothest Genes		**Least Smooth Genes**	
1.*Zfp931*	2.*Cntnap5c*	1.*Tmsb10*	2.*Slc25a4*
3.*Gm11508*	4.*Pm20d2*	3.*Rps23*	4.*Rpl17a*
5.*Atp6ap1*	6.*Frmd3*	5.*Rpl37a*	6.*Rpl37*
7.*Dnah8*	8.*Rbms3*	7.*Rps11*	8.*Rps27*
9.*4930426I24Rik*	10.*Gm12089*	9.*Rps24*	10.*Rps16*

Table 6.5: *Top 10 smoothest and least smooth genes after running the Gaussian Process with Composite Kernel model on the Dentate Gyrus dataset.*

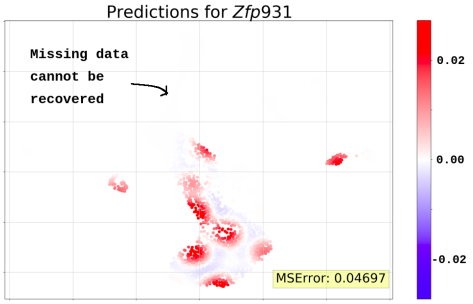
Gaussian Process Model with RNA Velocity Observations			
Smoothest Genes		**Least Smooth Genes**	
1.*Gm11508*	2.*Cntnap5c*	1.*Tmsb10*	2.*Rps23*
3.*Zfp931*	4.*Atp6ap1*	3.*Slc25a4*	4.*Rpl37a*
5.*Frmd3*	6.*Gm12089*	5.*Rpl17*	6.*Rpl37*
7.*4930426I24Rik*	8.*Pm20d2*	7.*Rps27*	8.*Rps16*
9.*Ticam1*	10.*A430105J06Rik*	9.*Rps24*	10.*Rps11*

Table 6.6: *Top 10 smoothest and least smooth genes after running the Gaussian Process Model with RNA Velocity observations on the Dentate Gyrus dataset.*

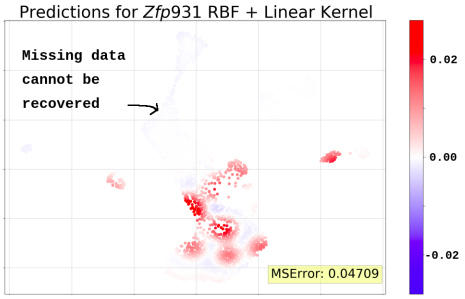
The ranking of the Baseline model and the model with RNA Velocity observations is consistent across all genes. Nevertheless, the top ten least smooth genes are the same for all three models.

This observation again implies that the least smooth genes are carrying more information when the dataset is severely sparse (in this case ~ 77% are zeros).

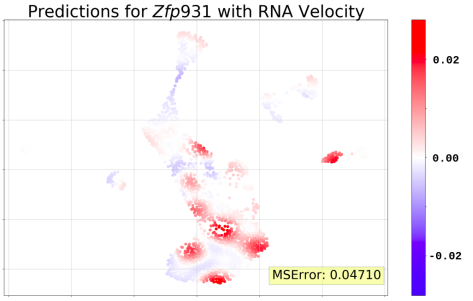
To examine the behaviour of the models when there are only a few reads of a gene, gene expression functions as by each model for the top smooth gene 'Zfp931' are shown in Figure 6.5



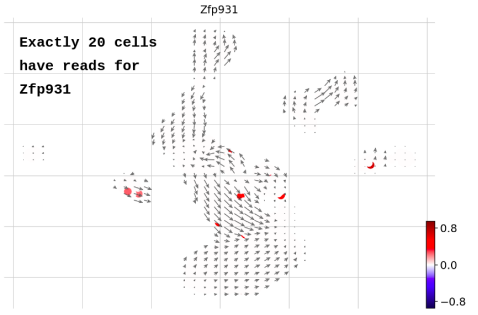
(a) The gene expression function predicted by the Baseline Gaussina Process.



(b) The gene expression function predicted by the GP with Composite Kernel.



(c) The gene expression function predicted by the GP with RNA Velocity.



(d) The true values of the gene expression including the RNA velocity.

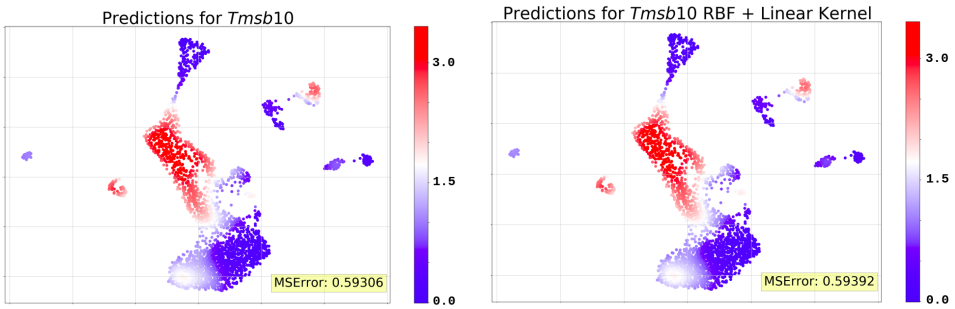
Figure 6.5: *The predicted gene expression function for the 'Zfp931' gene. (d) shows the RNA Velocity UMAP low dimensional map obtained by the 'stochastic' scvelo model. For all four graphs there are a lot of 0s both in the predictions and in the true values.*

As seen from Figure 6.5d the number of reads at which the intensity of the gene signal is visible are extremely low. The exact count of non-zero values for 'Zfp931' is 20 (> 1% of cells)[1]. Thus, when sampling even at 70% of the data, if all datapoints are 0s the model predicts 0s everywhere. This is seen in Figure 6.5a and Figure 6.5b. The effect of adding the RNA Velocity is evident in Figure 6.5c. The RNA Velocity smooths the gene expression function and assigns values even when they are not present in the initial observations as opposed to the baseline model and the model with composite kernel. This assignment of low intensity values results in a higher mean squared error for the model with RNA Velocity observations.

The top genes look more like outliers as they are expressed in only a small number of cells. Thus, the behaviour of the least smooth genes ranked by the models is shown in Figure 6.6

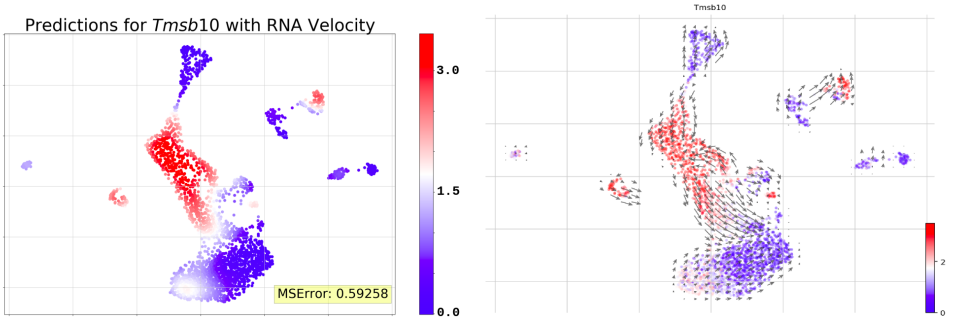
As seen from Figure 6.6 all three models are very good fit to the dataset. However, the Gaussian Process model with RNA Velocity observations results in the smallest mean squared error. Such behaviour was evident in the results from subsection 6.2.2. Therefore, the RNA Velocity observations are improving the Gaussian Process model for the smoothest genes when the data is

[1] Evaluating such genes is tricky as they could be low quality reads (technical noise) or cell's property (biological noise).



(a) The gene expression function predicted by the Baseline Gaussina Process.

(b) The gene expression function predicted by the GP with Composite Kernel.



(c) The gene expression function predicted by the GP with RNA Velocity.

(d) The true values of the gene expression including the RNA velocity.

Figure 6.6: *The predicted gene expression function for the 'Tmsb10' gene labeled as the least smooth gene by all three models. (d) shows the RNA Velocity UMAP low dimensional map obtained by the 'stochastic' scvelo model.*

not too sparse. Nevertheless, when the gene expression reads are very sparse, the RNA Velocity can be used as a reconstruction technique which is useful when drop-outs are present.

Performing a paired two-tailed t-test at 95% confidence on the mean square errors from the Baseline Gaussian Process model vs the Gaussian Process model with RNA Velocity observations for each gene also shows significant difference between the two models. The Gaussian Process model with RNA Velocity again has smaller mean (t-value is negative), thus better fit. The value of t is -2.233128. The value of p is 0.02637. The result is significant at $p < .05$.

6.3 Models Evaluation

To summarize, Gaussian process models are very powerful for identifying smoothly varying genes across cells in an unsupervized manner. Depending on the desired smoothness, different kernel functions can be used and hyperparameters can be tweaked to make the model better fit to the data (see Table 6.7). For this particular implementation, the marginal likelihood is obtained analytically leveraging faster computations and optimization including derivatives. In addition, the likelihood is used as a criterion for smoothness allowing unsupervized ranking of genes. The Gaussian process model can be particularly customized for scRNA-seq datasets as the prior covariance matrix needs be computed only once per dataset.

Furthermore, RNA Velocity information can be incorporated as derivative observations. This additional information allows less cells to be used in the computations provided that the dataset

is not extremely sparse. RNA Velocity observations improve the models fit and can result in re-ranking of the genes.

In the case when the dataset has a lot of 0 reads, the RNA Velocity can be used as a tool for data reconstruction. However, if the low intensity reads are a cell property, RNA Velocity will be at disadvantage as it will smooth the gene expressions function and will lead to incorrect values. In addition, when gene intensity levels are low and global velocity is high, scaling constant is desired.

The covariance function can incorporate different behaviours as implemented in the Gaussian process model with composite kernel (comprising the sum of Linear and Squared Exponential kernels) and result in different family curves. Such behaviour can approximate derivative observations if the derivatives are consistent with the slopes of neighbouring cells, however, it does not substitute the new information introduced by RNA Velocity.

Finally, the Gaussian Process model with RNA Velocity observations results in smoother gene expression functions compared to the Baseline Gaussian Process model and the Gaussian Process with Composite Kernel model. However, there are some issues that might arise when using velocity. In low dimensional mapping, the velocity vector field is constant for cells, thus a scaling constant may be appropriate for specific genes but not all. Additionally, including RNA Velocity measurements may introduce noise rather than actual gene predictions when lots of zero counts are recorded in the count matrix. The definition of smoothness when the dataset is extremely sparse is biased towards genes with very low counts regardless of the chosen model. Nevertheless, ranking genes by their smoothness as defined by the Gaussian Process model can still reveal interesting properties of the genes regardless of the dataset.

	Baseline Gaussian Process	**Gaussian Process with Composite Kernel**	**Gaussian Process with RNA Velocity**
Advantages	1. Fast computations 2. Easily optimization of hyperparameters 3. Good starting point	1. More complex behaviour of the curve families 2. Fast computations	1. Less observations are needed as new information is introduced 2. Smoother gene expression functions
Disadvantages	1. Tuning hyperparameters for smoother family curves may result in a too simple or too complex model	1. More hyperparameters to optimize	1. The kernel matrix size grows exponentially with velocity dimensions 2. Computations take much longer as observations increase 3. Memory increases exponentially with velocity dimensions 4. Scaling constant for velocity may be needed

Table 6.7: *Advantages and disadvantages of the three main models to be evaluated*

7 Conclusion

This chapter provides a summary of the problems and goals associated with gene profiling and gene ranking by smoothness in scRNA-seq datasets, the proposed solution and its limitations. Furthermore, there is a discussion of possible improvements and additional features to be added in the proposed solution.

7.1 Problem Scope

This dissertation focuses on the use of scRNA-seq datasets in combination with Gaussian Processes and RNA Velocity for the profiling and unsupervized ranking of genes by their smoothness. Smoothly varying genes are of particular interest because they could be a direct consequences of cellular differentiation events as suggested by EL Low (2019). In addition, the identification of such genes can unravel hidden structures in the dataset and describe biological properties of the system. Gaussian Process are used particularly because of their intrinsic smoothness definition and the ability to further incorporate RNA Velocity as derivative observations.

7.2 Solution and Results

For this particular problem Gaussian Processes are used due to the properties of the model's covariance functions which define a very smooth latent space in the calculus sense. In addition, Gaussian Processes enable derivative observations as additional information. This allows RNA Velocity vector to be integrated in the original model.

Specifically, Gaussian Processes can integrate the information for the derivatives at the observation points, thus allowing RNA Velocity to be added as derivative for the gene expressions in cells. This new information results in smoother fit for most genes and generally better results. In addition, when the observations are very sparse, the RNA Velocity ca be used as a tool for reconstruction of the data. However, this can be misleading as the sparse gene expression levels might be properties of the cells. This means the reconstruction is just introduced noise and does not present the actual observations. In addition, when the dataset is too sparse, the model is biased (identifies as smoothest) towards genes with extremely low intensity levels.

The implementation of the Gaussian Process class for scRNA-seq datasets proposed in this dissertations runs 150 times faster than other libraries implementing Gaussian Processes. This effect is due to the cashing of the covariance matrix. Furthermore, the implementation is compatible with existing tool sets, e.g. `sklearn`.

7.3 Future Improvements

The feasibility of using Gaussian process to determine smooth gene has been demonstrated. However, further improvements can additionally enhance the power of the method and enable biologist to make new findings.

The model does not implement the full derivative of the marginal likelihood w.r.t. the hyperparameters when the kernel function incorporates derivative observations. These derivatives can be analytically obtained. Thus, the implementation should be trivial and the full derivatives can be easily added.

In Figure 6.4 it is shown how the RNA Velocity rapidly boosts the prediction values of gene expression functions at regions with high velocity levels which results in incorrect predictions for the gene expression. Therefore, adding a scaling constant for the RNA Velocity and further optimizing it for the specific genes can lead to better results.

The current implementation of the Gaussian process assumes Gaussian likelihood of the initial observations. However, for scRNA-seq datasets this is usually not the case due to the non-negative counts and the high amount of drop-outs (many 0s counts). Thus, a model incorporating such information may result in better fit of the dataset, thus better rankings.

Communicating the results to biologists can give another insight of what information the found genes carry and what biological meaning they have. Furthermore, it can popularize the method and give rise to a discussion about other gene features which can be interesting for the scRNAseq datasets and can be obtained with Gaussian Processes.

A Appendices

A.1 Human Saphenous Vein Dataset: Normal Batch

The results from running the three models on the human saphenous vein dataset normal batch are summarized below.

Baseline Gaussian Process Model			
Smoothest Genes		**Least Smooth Genes**	
1.*LMO7*	2.*TXNRD1*	1.*KYNU*	2.*ST6GALNAC5*
3.*GLS*	4.*TNFRSF11B*	3.*PCDH7*	4.*TMEFF2*
5.*FST*	6.*TENM4*	5.*IL1A*	6.*LINC01705*
7.*ANKH*	8.*SLC14A1*	7.*DRAXIN*	8.*AFF2*
9.*CCDC80*	10.*CALD1*	9.*LINC02154*	10.*LINC01423*

Table A.1: *Top 10 smoothest and least smooth genes after running the Gaussian Process with Composite Kernel model on the Human Saphenous Vein Dataset: Normal Batch.*

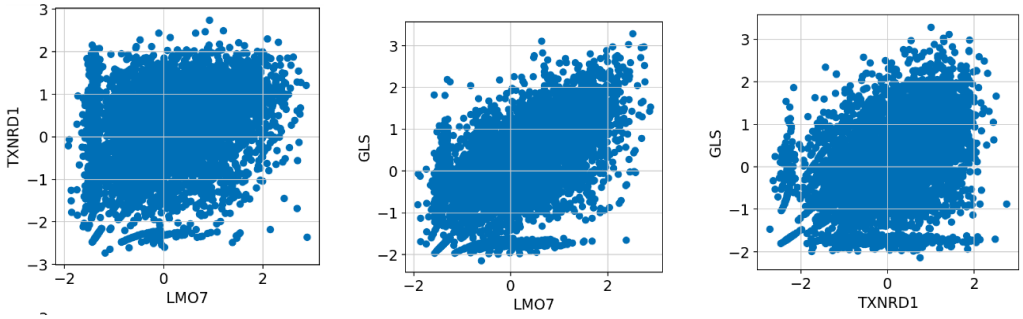
Gaussian Process Model with Composite Kernel			
Smoothest Genes		**Least Smooth Genes**	
1.*LMO7*	2.*TXNRD1*	1.*ST6GALNAC5*	2.*DRAXIN*
3.*TNFRSF11B*	4.*ANKH*	3.*CADM1*	4.*EFNB2*
5.*CALD1*	6.*GLS*	5.*KCNJ3*	6.*IL1B*
7.*FGF2*	8.*TENM4*	7.*PNP*	8.*PURPL*
9.*NEK7*	10.*CCDC80*	9.*LINC02154*	10.*BMF*

Table A.2: *Top 10 smoothest and least smooth genes after running the Gaussian Process Model with RNA Velocity observations on the Human Saphenous Vein Dataset: Normal Batch.*

Gaussian Process Model with RNA Velocity Observations			
Smoothest Genes		**Least Smooth Genes**	
1.*LMO7*	2.*FGF2*	1.*ST6GALNAC5*	2.*CHRM3*
3.*TNFRSF11B*	4.*CALD1*	3.*NPY1R*	4.*LINC02257*
5.*TXNRD1*	6.*GLS*	5.*ANKRD45*	6.*LINC01013*
7.*ANKH*	8.*FST*	7.*DRAXIN*	8.*STK32A*
9.*SLC14A1*	10.*GRAMD2B*	9.*PRKG2*	10.*TMEM132B*

Table A.3: *Top 10 smoothest and least smooth genes after running the Gaussian Process Model with RNA Velocity observations on the Human Saphenous Vein Dataset: Normal Batch.*

The correlation between three of the smoothest genes similarly identified by all three models is shown in Figure A.1



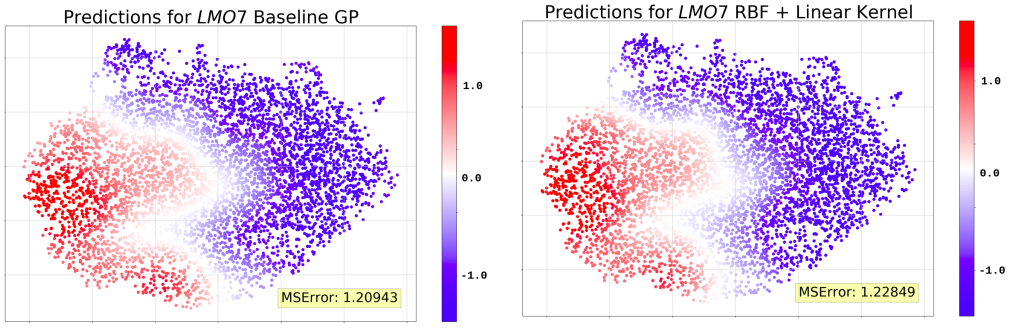
(a) Correlation between gene LMO7 and TXNRD1.

(b) Correlation between gene LMO7 and GLS.

(c) Correlation between gene TXNRD1 and GLS.

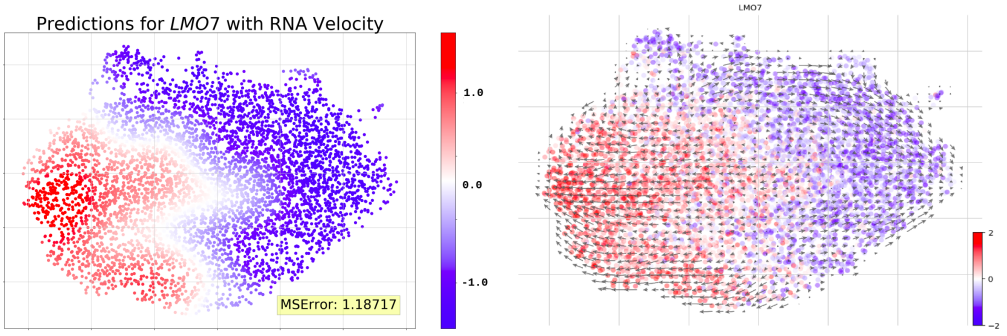
Figure A.1: *The correlation from the top 3 smoothest genes as discovered by the baseline Gaussian process model. There is a clear correlation between the smoothest genes.*

Example expression function as derived by all three models for the 'LMO7'gene is shown in Figure A.2



(a) The gene expression function predicted by the Baseline Gaussian Process.

(b) The gene expression function predicted by the GP with Composite Kernel.



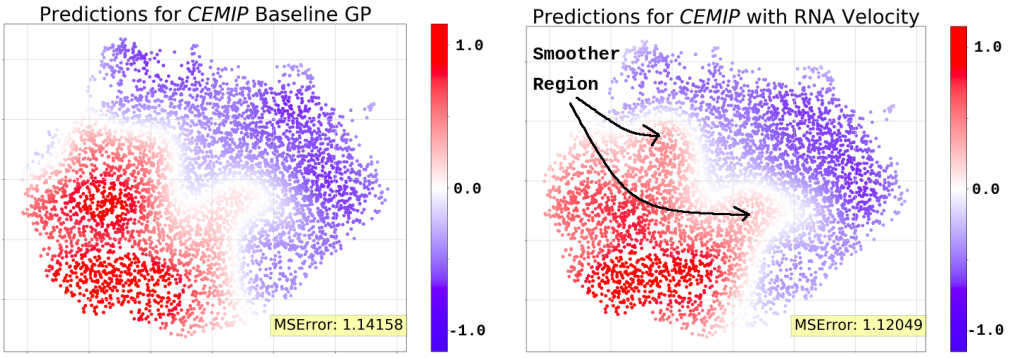
(c) The gene expression function predicted by the GP with RNA Velocity.

(d) The true values of the gene expression including the RNA velocity.

Figure A.2: *The predicted gene expression function for the 'LMO7' gene. (d) shows the RNA Velocity UMAP low dimensional map obtained by the 'stochastic' scvelo model.*

As seen from Figure A.1 the Gaussian Process with RNA Velocity observations has the best accuracy and the smoothest fit.

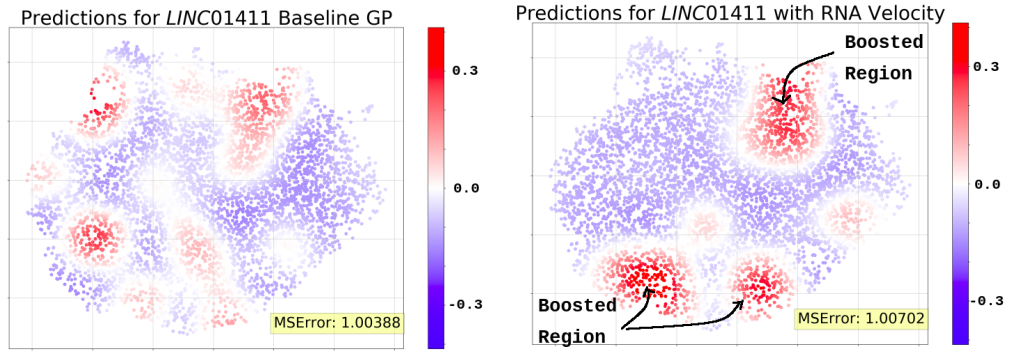
Re-ranking of top 50 genes (found by Baseline Gaussian Process Model) as a consequence of RNA Velocity and the effect on the gene expression functions shown in Figure A.3 and Figure A.4.



(a) The gene expression function for 'CEMIP' predicted by the Baseline Gaussian Process.

(b) The gene expression function for 'CEMIP' predicted by the GP with RNA Velocity.

Figure A.3: *The predicted gene expression functions for the 'CEMIP' which has been ranked 18 places higher by the GP with RNA Velocity model. The model in (a) keeps the variations in the gene expression while the model in (b) smooths the gene at the shown regions due to velocity levels.*



(a) The gene expression function for 'LINC01411' predicted by the Baseline Gaussian Process.

(b) The gene expression function for 'LINC01411' predicted by the GP with RNA Velocity.

Figure A.4: *The predicted gene expression functions for the 'LINC01411' which has been ranked 45 places lower by the GP with RNA Velocity model. The model in (a) keeps the low gene expression values while the model in (b) boosts those values where velocity levels are high and lowers them where velocity levels are low.*

7 Bibliography

S. J. B. T. e. a. Eberwine, J. The promise of single-cell sequencing. *Nat Methods*, 11:25–27, 2014. doi: https://doi.org/10.1038/nmeth.2769.

A. K. M. P. D. K. A. S. M. T. J. M. M. M. S. A.-R. S. F. N. M. C. D. P. H. M. H. S. N. P. T. D. A. B. A. B. EL Low, JT Schwartze. Conserved properties of dentate gyrus neurogenesis across postnatal development revealed by single-cell rna sequencing. *bioRxiv*, 2019. doi: https://doi.org/10.1101/860320.

Z. A. L. P. e. a. Hochgerner, H. Conserved properties of dentate gyrus neurogenesis across postnatal development revealed by single-cell rna sequencing. *Nat Neurosci*, 21:290–299, 2018. doi: https://doi.org/10.1038/s41593-017-0056-2.

S. R. Z. A. e. a. La Manno, G. Rna velocity of single cells. *Nature*, 560:494–498, 2018. doi: https://doi.org/10.1038/s41586-018-0414-6.

J. M. Leland McInnes, John Healy. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv*, 2018.

Z. C. V. B. e. a. Parekh, S. The impact of amplification on differential expression analyses by rna-seq. *Sci Rep*, 6:25533, 2016. doi: https://doi.org/10.1038/srep25533.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. ISBN 0-262-18253-X.

M.-S. R. L. W. R. C. Solak, E. and D. Leith. Derivative observations in gaussian process models of dynamic systems. *Neural Information Processing (NIPS) Meeting*, 2002.

B.-C. W. Y. e. a. Tang, F. mrna-seq whole-transcriptome analysis of a single cell. *Nat Methods*, 6:377–382, 2009. doi: https://doi.org/10.1038/nmeth.1315.

P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. . . Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17: 261–272, 2020. doi: https://doi.org/10.1038/s41592-019-0686-2.

S. P. F. A. W. F. J. T. Volker Bergen, Marius Lange. Generalizing rna velocity to transient cell states through dynamical modeling. *bioRxiv*, 2019. doi: https://doi.org/10.1101/820936.

A. P. . T. F. Wolf, F. Scanpy: large-scale single-cell gene expression data analysis. *Genome Biol*, 19:15, 2018. doi: https://doi.org/10.1186/s13059-017-1382-0.