Gaussian Processes for Time-Varying Treatment Effects

Gaussian Processes for Time-Varying Treatment Effects

A DISSERTATION PRESENTED
BY
JUSTIN ZHU
TO
THE DEPARTMENT OF COMPUTER SCIENCE AND THE DEPARTMENT OF MATHEMATICS

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
BACHELOR OF ARTS
IN THE JOINT SUBJECTS OF
COMPUTER SCIENCE AND MATHEMATICS

HARVARD UNIVERSITY
CAMBRIDGE, MASSACHUSETTS
MAY 2021

Thesis advisor: Professor Susan Murphy Justin Zhu

Gaussian Processes for Time-Varying Treatment Effects

Abstract

Gaussian Process (GP) models have gained popularity for their flexibility to handle correlation among data sampled from the Gaussian Distribution. The correlation frequently characterizes time-dependent data, such as step count data across different time-horizons. Unlike Gaussian Processes used in a regression setting, the outcomes y are not continuous but are discrete, and the posterior distribution of the outcomes conditioned on our data cannot be solved analytically in closed-form.

This thesis surveys the theory and computation of Gaussian Processes in fitting time-varying treatment effects, drawing upon binary outcomes for its simplicity in illustrating the ideas behind fitting Gaussian Processes on non-Gaussian parameters. Binary outcomes are defined by a Bernoulli distribution, from which the probability parameter π exists as both a function of input variables as well as a random variable described by a Gaussian Process.

Gaussian Processes fitted on changing values of π over time provide greater flexibility in describing the effects of input variables on a target variable across time. This flexibility has certain desirable qualities in context of causal inference, decision theory, and bandit learning.

This is dedicated to everybody I had the privilege of meeting in my 21 years so far – and the many more who have yet to come.

Acknowledgments

I would like to thank my thesis adviser Professor Susan Murphy for the kindness and knowledge she's given me all this time, from when I first entered her lab as a sophomore in college to the years now beyond. While there is so much to thank her for, I am perhaps most inspired by her ever-present enthusiasm for investigating difficult problems, her generosity in teaching others, and her sense of responsibility for how her research is to make the world a better place. Researching with her lab has been the privilege of my undergraduate career.

I would also like to thank Dr. Tianchen Qian for the mentorship he's given me when I first started in Susan's lab. Tianchen's understanding of statistics is some of the deepest I've seen, and I would be remiss to not credit him with providing me the statistical foundations for this thesis and my research as a whole.

Along the way, I have been blessed by the pedagogy of Professor Joe Blitzstein, whose two classes on probability have provided some of the most elegant and intuitive explanations of probability I've experienced, and many others – Professors Eddie Kohler and James Mickens for their teachings on systems programming, Professor David Parkes for his insightful course on economics and computation, Professor Lucas Janson for his teachings on statistical inference and learning, and Professor Finale Doshi-Velez for her support in machine learning research.

It takes a village to raise a child, and just so, it takes a society to raise a student. So many people have sacrificed so much just so I could continue my studies in a post-covid world. To all of them, I am forever grateful.

Contents

1

Introduction

Change is the only constant in life.

This observation, made by the Greek philosopher Heraclitus over 2500 years ago, has never seemed more appropriate in our lives today. The 21st century has been defined by constant change, not only in the physical world but even more so in a digital one, as the Internet has permeated every nook and cranny of our everyday lives. Now marks a time where granular details in all our changing activities can be stored by big-data systems and processed by efficient algorithms. The rise of fields like machine learning, cloud computing, and reinforcement learning further perpetuate this understanding that the changes in our lives can be studied and engineered in radically new ways.

As change happens so frequently in our lives, how is this change reflected in the data that we collect? We can quantify this change by using *mathematical models*, which are formulas that describe relationships between variables that have crystallized into observed data. While the Gaussian Process model is a popular model for describing change, there is an abundance of other models that can also describe changes over time. The difference between models can often be attributed to different *assumptions* about the data, which are our beliefs on how the data should behave.

Once we have formulated a model, we would like to find ways of testing this model. Computational tools allow us to test models by simulating data or splitting data, often randomly, thereby creating a *testbed*. Using our testbed that we have generated computationally, we can then evaluate the *performance* of our model.

Performance can have many definitions, but in context of a model, it refers to how well

the model performs with respect to a *loss function*, which is a function of the difference between the estimated values of the testbed data as predicted from our model and the true values from the testbed data. In general, the lower the values we obtain from our loss function after plugging in our predicted values from our model, the better the performance. It is a convention to often use loss function and *objective function* interchangably despite the difference in meaning – the loss function can be thought of a as negative objective function so that optimizing the loss function means minimizing the loss function and optimizing the objective function means maximizing the objective function.

Performance also has another definition used more prevalently in computer science, which is how well computation scales as a function of the data. The less computations needed in our model for every incremental increase of data, the better the performance. This type of performance is also known as *computational complexity*, and is described using a big-O notation ($\mathcal{O}(n)$) where n is typically a proxy of data quantity.

Computation provides rich insights into ways we can better optimize and refine our models so that we can formulate more models that have even better performance, according to both the loss function and computational complexity. Through this constant iteration between formulating models and evaluating our models computationally, we can ideally come close to creating a perfect model that describes the change that constantly enters our lives.

2

Gaussian Processes

Gaussian Processes are a class of *stochastic processes*[1], which are models that describe outcomes that behave randomly. The opposite of a stochastic process is a deterministic process, where outcomes do not behave randomly. While deterministic processes are fixed and can be better studied using the tools of pure mathematics, stochastic processes are flexible and can be studied using the tools of computation.

An example of a deterministic process is Newton's law of gravity. Because gravity's acceleration is fixed to be 9.8 meters per second squared according to Newton's law, we can use this acceleration to calculate the time it takes an object to hit the ground from a certain height, guaranteeing this time value will never change. In contrast, a stochastic process could characterize a particle moving in air. We can never guarantee where the particle be at any given moment in time, but we can model the stochastic behavior of the particle by describing the probability the particle will be in a given space at any given moment in time.

Many real-life phenomenon behave stochastically rather than deterministically, which motivates the study and formalization of stochastic processes.

[1] The word stochastic is originally derived from the Greek word *stok-hazesthai*, which means to aim or to guess at. Today, it's used more commonly as a synonym for random.

2.1 Stochastic Processes

A stochastic process is a collection of random variables $\{X_t, t \in I\}$ The set I is the index set of the processes, indexed by t, which typically represents time. The random variables

are defined on a common state space S. When describing a stochastic process, the random variables[2] across state and time are of particular interest.

[2] A random variable is formally defined as a function that maps objects into a number between 0 and 1.

Time

The time t can take on discrete or continuous values. If t is continuous, we write mathematically $t \in [0, \infty)$ and if t is discrete, we write mathematically, $t \in \{0, 1, 2, \cdots\}$. Time is an unbounded variable and therefore approaches infinity.

A Gaussian Process has continuous values of t while a Markov Process has discrete values of t. The implications of discrete and continuous values will become apparent in later chapters.

State Spaces

The state space, represented as S, represents the state of values X_t can take on at any given point in time. A Gaussian Process has continuous state space $S = (-\infty, \infty)$ while a Markov Process has countable state space $S = \{0, 1, 2, \cdots, N\}$.

2.2 Gaussian Process Definition

A Gaussian process $(X_t)_{t \geq 0}$ is a continuous-time stochastic process with the property that for all $n = 1, 2, \ldots$ and $0 \leq t_1 < \cdots < t_n$, the random variables $X_{t_1}, \ldots, X_{t_n}$ have a multivariate normal distribution. Because $X_{t_1}, \ldots, X_{t_n}$ have a multivariate normal distribution, any instance of X_t follows a normal[3] distribution.

[3] The Normal distribution and the Gaussian distribution refer to the same distribution: $\mathcal{N}(\mu, \sigma^2)$

2.3 Multivariate Distribution

A multivariate distribution normal distribution is a collection of Gaussian, or Normal, random variables $\mathbf{X} = (X_1, X_2, X_3, \cdots X_k)^T$[4].

$$\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

[4] A convention used here is to let bold symbols denote vectors or matrices of random variables, while

where $\boldsymbol{\mu} = (\mu_1, \cdots, \mu_k) \in \mathbb{R}^k$ and $\boldsymbol{\Sigma} = \begin{pmatrix} \sum_{11} & \cdots & \sum_{1k} \\ \vdots & \ddots & \vdots \\ \sum_{k1} & \cdots & \sum_{kk} \end{pmatrix} \in \mathbb{R}^{k \times k}$.

Mean Vector

A mean vector is a $p \times 1$ vector, where p denotes the number of features in our dataset[5]. We write the sample mean vector as a column vector.

$$\overline{\mathbf{x}} = \begin{bmatrix} \bar{x}_1 \\ . \\ . \\ \bar{x}_p \end{bmatrix}$$

[5] p or k is typically used to denote dimensionality of features while n is used to denote number of observations.

Note that the sample mean vector is different from the population mean vector, which we write also as a column vector.

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ . \\ . \\ \mu_p \end{bmatrix}$$

The difference between sample mean and population mean is subtle. The sample mean is a *statistic* that is calculated, while the population mean is a *parameter* that is estimated. The only way we know our population mean is often when we set it ourselves in a testbed – rarely is that provided to us in real life.

Being able to identify the true parameter as a function of a statistic is called *inference* in the field of statistics, or *learning* in the field of computer science.

Law of Large Numbers

The law of large numbers help relate our sample mean statistic and population mean parameter. It states that the sample mean $\bar{x}$ approaches the population mean μ with probability 1 as the number of observations increase.

Proof:

We can use Chebyshev's inequality which states that for any random variable X and for all $\varepsilon > 0$,

$$P(|X - E[X]| \geq \epsilon) \leq \frac{\text{Var}(X)}{\epsilon^2}$$

We calculate the expected value of our random variable $\bar{X}_n$, where $\bar{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i$.

$$\begin{aligned} E\left[\bar{X}_n\right] &= E\left[\frac{1}{n}\sum_{i=1}^{n} X_i\right] \\ &= \frac{1}{n}\sum_{i=1}^{n} E\left[X_i\right] \\ &= \frac{1}{n}\sum_{i=1}^{n} \mu \\ &= \frac{n\mu}{n} \\ &= \mu \end{aligned}$$

We procced to apply similar rules to calculate the variance of our random variable $\bar{X}_n$

$$\begin{aligned} \text{Var}\left(\bar{X}_n\right) &= \text{Var}\left[\frac{1}{n}\sum_{i=1}^{n} X_i\right] \\ &= \frac{1}{n^2}\text{Var}\left(\sum_{i=1}^{n} X_i\right) \\ &\overset{iid}{=} \frac{1}{n^2}\sum_{i=1}^{n} \text{Var}\left(X_i\right) \\ &= \frac{1}{n^2}\sum_{i=1}^{n} \sigma^2 \\ &= \frac{n\sigma^2}{n^2} \\ &= \frac{\sigma^2}{n} \end{aligned}$$

We are able to make the substitution $\frac{1}{n^2}\text{Var}\left(\sum_{i=1}^{n} X_i\right) = \frac{1}{n^2}\sum_{i=1}^{n}\text{Var}\left(X_i\right)$ because the random variables $X_1, \cdots, X_n$ are independent[6] to one another.

Now substitute values of $E\left[\bar{X}_n\right]$ and $\text{Var}\left(X_n\right)$ into Chebyshev's Inequality to obtain

$$P\left(\left|\bar{X}_n - \mu\right| \geq \epsilon\right) \leq \frac{\sigma^2}{n\epsilon^2}$$

[6] We say two variables are independent to one another if $P(X \text{ and } Y) = P(X)P(Y)$

It is clear that as as number of data points increase, $n \to \infty$ so that $\lim_{n\to\infty} P\left(\left|\bar{X}_n - \mu\right| \geq \epsilon\right) = 0$, illustrating that sample mean $\bar{X}$ converges in probability to μ.

Consistency

We say that a statistic is *consistent* if it converges to the parameter in probability as the number of observations increase. Under the law of large numbers, we have shown that the sample mean $\bar{X}$ is a consistent estimator of μ.

Covariance Matrix

In addition to the mean vector, the covariance matrix is the additional parameter characterizing the Multivariate Normal distribution. We can write out the covariance matrix to store covariances between random variables X_i and X_j in the (i, j) position, represented mathematically as Σ_{ij} for values of $i \in \{1, \cdots, k\}$ and $j \in \{1, \cdots, k\}$.

$$\Sigma = \begin{bmatrix} \Sigma_{11} & \cdots & \Sigma_{1k} \\ \vdots & \ddots & \vdots \\ \Sigma_{k1} & \cdots & \Sigma_{kk} \end{bmatrix}$$

Covariance Function

For any two variables X_i and X_j, the covariance function between them is the expected product minus the product of expectations.

$$\begin{aligned} \Sigma_{ij} &= \text{Cov}(X_i, X_j) \\ &= E((X_i - E(X_i)(X_j - E(X_j))) \\ &= E(X_i X_j - E(X_i)X_j - X_i E(X_j) + E(X_i)E(X_j)] \\ &= E(X_i X_j) - E(X_i)E(X_j) - E(X_i)E(X_j) + E(X_i)E(X_j) \\ &= E(X_i X_j) - E(X_i)E(X_j) \end{aligned}$$

When $i = j$, i.e., the random variables X_i and X_j become two identical random variables, the covariance becomes variance.

$$\begin{aligned}\Sigma_{ii} &= \text{Cov}(X_i, X_i) \\ &= E(X_i X_i) - E(X_i)E(X_j) \\ &= E(X_i^2) - E(X_i)^2 \\ &= \text{Var}(X_i)\end{aligned}$$

From this observation relating covariance and variance, we can conclude that the diagonal along the covariance matrix is always the variance of a particular i-th random variable. Thus, we can write out all the terms in our covariance matrix as either a variance of a single random variable or the covariance of two different random variables.

$$\mathbf{\Sigma}_{ij} = \begin{cases} \text{Var}(X_i) & \text{if } i = j \\ \text{Cov}(X_i, X_j) & \text{if } i \neq j \end{cases}$$

A key point to be made here is the difference between *covariance matrix* and *covariance function*. The covariance function is a function of *random variables*, while the covariance matrix is a matrix of *observations*. Recall that we can think of observations in our data as crystallized instances of random variables – in other words, when a random variable ceases to be random anymore, it becomes an observation. Likewise, a covariance matrix is a crystallization of our covariance function. When random variables become numbers, a covariance function of random variables becomes a number as well.

While the word "covariance" on its own is frequently used in scientific literature without this distinction between covariance function or covariance matrix, it is almost always implied that covariance refers to the covariance matrix or a value inside the covariance matrix. This is because rarely do covariance functions change from the definition $\text{Cov}\left(X_i, X_j\right) = E\left(X_i X_j\right) - E\left(X_i\right) E\left(X_j\right)$ unless there exists compelling reason to do so. Therefore, because the function is not of interest but the values are, covariance generally refers to the covariance matrix.

However, in this dissertation, we are interested in examining changing definitions of the *covariance function* because changing definitions of the covariance function create greater generalizations of the Gaussian Process model. To lessen any possible ambiguity between the covariance matrix and the covariance function, we introduce the word *kernel* to mean the same thing as the *covariance function*, and *covariance* as a shorthand for covariance matrix or a value inside the covariance matrix.

Probability Density Function

A distribution is characterized by a probability density function (PDF), which expresses the infinitesimal probability mass around any given point in a continuous distribution.

When the density function is integrated over the entire support[7] of values, the entire integral sums to 1. The PDF will always output a nonnegative value.

[7] The support is the range of values that our random variable can take on, i.e., the sample space.

As an example, the probability density function of the normal distribution is

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

The probability density function of the multivariate normal distribution follows a similar convention when written using vector notation.

$$f(\boldsymbol{X}) = \frac{1}{(2\pi)^{k/2}|\det(\boldsymbol{\Sigma})|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$$

Of particular interest is the determinant function, present in the PDF of the multivariate normal distribution but not in the PDF of the single variable distribution. The most intuitive explanation of the determinant is the volume of our multivariate distribution stretched out in a multidimensional space, where the dimension of the multidimensional space is the number of random variables in our multivariate distribution.

2.4 Properties of Multivariate Normal

The Multivariate Normal Distribution has a few elegant properties that allow us to derive solutions analytically, with minimal use of computational tools.

Linear Combinations of Normal Distributions

A linear combination of Gaussian random variables is a Gaussian random variable[8].

[8] The formal proof involves showing characteristic functions $\varphi_X(t) = \mathrm{E}\left[\exp\left(it^T X\right)\right]$ to be equal. Characteristic functions are Fourier transforms

Let Y be the linear combination of p normal random variables $X_1, \cdots, X_p$. Note that $(X_1, X_2, \cdots, X_p)^\top$ form a multivariate normal distribution.

$$\begin{aligned} Y &= c_1 X_1 + c_2 X_2 + \cdots + c_p X_p \\ &= \sum_{j=1}^{p} c_j X_j \end{aligned}$$

Y itself would form a normal distribution with certain mean and certain variance parameters.

The mean of Y can be calculated in a straightforward manner using the linear combinations of $X_1, \cdots, X_p$.

$$\begin{aligned} E(Y) &= E\left(\sum_{j=1}^{p} c_j X_j\right) \\ &= \sum_{j=1}^{p} c_j E(X_j) \\ &= \boldsymbol{c}^\top \boldsymbol{\mu} \end{aligned}$$

where

$$\boldsymbol{\mu} = \begin{bmatrix} E(X_1) \\ E(X_2) \\ \vdots \\ E(X_p) \end{bmatrix}, \quad \boldsymbol{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_p \end{bmatrix}$$

The variance of Y can be calculated using the covariance matrix $\boldsymbol{\Sigma}$ of

$(X_1, X_2, \cdots, X_p)^\top$, a multivariate normal distribution.

$$
\begin{aligned}
\text{Var}(Y) &= \text{Cov}(Y, Y) \\
&= \text{Cov}\left(\sum_{j=1}^{p} c_j X_j, \sum_{j=1}^{p} c_j X_j\right) \\
&= E\left(\left(\sum_{j=1}^{p} c_j X_j\right)^2\right) - E\left(\sum_{j=1}^{p} c_j X_j\right)^2 \\
&= E\left(\sum_{j=1}^{p} c_j^2 X_j^2 + 2\sum_{i<j}^{p} c_i c_j X_i X_j\right) \\
&- \left(\sum_{j=1}^{p} c_j^2 E(X_j)^2 + 2\sum_{i<j}^{p} c_i c_j E(X_i) E(X_j)\right) \\
&= \sum_{j=1}^{p} c_j^2 (E(X_j^2) - E\left(X_j\right)^2) \\
&+ 2\sum_{i<j}^{p} c_i c_j \left(E(X_i X_j) - E\left(X_i\right) E\left(X_j\right)\right) \\
&= \sum_{j=1}^{p} c_j^2 \text{Var}(X_j) + 2\sum_{i<j}^{p} c_i c_j \text{Cov}(X_i, X_j) \\
&= \boldsymbol{c}^\top \boldsymbol{\Sigma} \boldsymbol{c}
\end{aligned}
$$

2.5 Assumptions

In theory, once we know the mean and the variance of our random variables $X_1, \cdots, X_n$ and the appropriate linear combination of $X_1, \cdots, X_n$ to obtain Y, our model is complete. However, in real life, rarely are we given the true behavior of random variables $X_1, \cdots, X_n$, and whether they follow the normal distribution at all!

In using a Gaussian distribution to describe a random variable, we've already created an *assumption* about our data – that the true behavior of our data follows the Normal distribution. *In fact, all models are defined by their assumptions about the data.* Assumptions sound unscientific, as if we're bringing personal bias into our examination of the data, but they are in fact quite necessary. If there were no assumptions about our data, it would be

impossible to create a model because there would be no structure in how we think about the data. This would be akin to writing a mathematical proof without any axioms to start with!

Of course, the less assumptions we make about our data, the more *generalizable* our model behaves, that is, the better our model will be able to describe relationships as new data arrives. This relates to a common topic of *overfitting*, where too much assumptions of our data cause the model to behave narrowly well on a particular existing dataset, but fail to capture the overall behavior of data. The goal of any good model should be to reduce *overfitting* so that its description of the data is as generalizable as possible.

2.6 Inference and Learning

Because we cannot observe the ground truth of how our data is generated, we must create models that come close in describing the ground truth of how our data is generated. Simply described, *inference* is the process of using observed data to describe a ground truth. We noted previously that *learning* is the same thing as inference, and is a term more commonly used in the vernacular of computer science. *Machine Learning*, then, is inference conducted by machines.

Machine Learning

What does it mean to have inference conducted by machines? On a very high level, it means to ultimately have machines identify the assumptions that go into a model. Traditionally, the standard workflow of developing a model has been:

1. Human constructs a model using assumptions
2. The machine evaluates model performance on some data testbed
3. Human refines model and assumptions based on results and data
4. Iterate between steps 2 and 3

Under machine learning, the third step could be automated away by machines:

1. Human constructs a model using assumptions
2. The machine evaluates model performance on some data testbed
3. The machine refines model and assumptions based on results and data
4. Iterate between steps 2 and 3

Note that in both cases, the initial step always starts with a human constructing an initial model with human assumptions. Assumptions are the axioms of any model, and as of date, machines are incapable of constructing their own axioms, although they are outstanding at using the axioms we've defined to identify new results via *algorithms*.

Machine Learning has emerged as its own category of learning in recent years mostly as consequence of better technology and more data. Computers have greater processing power than ever before, as the number of transistors on a chip now reach several billion. The more transistors on a chip, the more computations can be performed per unit time, enabling generalizable models that previously were computationally intensive to now be immediate, accelerating the iteration cycle between model evaluation and model refinement (steps 2 and 3). With great power in computation comes great power in inference.

Simultaneously, the rise of digital services have consolidated a wealth of data about everyday human activities on the computer and smartphone. We've previously seen how with the law of large numbers, the sample mean converges absolutely in probability to the population mean. Taken in a larger context, with a large number of data, a statistic converges absolutely in probability to a parameter. This implies that with more data, any parametric model approaches closer to the ground truth.

The same could be said of nonparametric models, which are a different class of models that do not rely on assumptions about the distribution of the model, the most popular example in machine learning literature being the neural network. Both nonparametric and parametric models have gained more power in recent years as a consequence of training on more data, leading to machines being able to accomplish extremely diverse tasks, leading to the popular conception of *artificial intelligence*, where the range of tasks that machines can perform have become so diverse that machines appear as capable as humans.

2.7 Design Matrix

The design matrix is the most common representation of data used in inference and learning.

Recall that we have a collection of random variables $X_1, \cdots, X_p$ that can be linearly combined to express a random variable Y. If we observe n observations of X_i for each $i \in \{1, \cdots, p\}$, then X_i can be written as a column vector of n elements:

$$X_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{in} \end{bmatrix}$$

If we were to concatenate all our X_i column vectors along the same row, we would have a design matrix. Here, the dimension of the design matrix is $n \times p$ (n rows and p columns):

$$\boldsymbol{X} = \begin{bmatrix} X_{11} & \dots & X_{1j} & \dots & X_{1p} \\ \vdots & & \vdots & & \vdots \\ X_{i1} & \dots & X_{ij} & \dots & X_{ip} \\ \vdots & & \vdots & & \vdots \\ X_{n1} & \dots & X_{nj} & \dots & X_{np} \end{bmatrix}$$

In our design matrix, X_{ij} is the i-th observation of the j-th variable or the j-th feature.

In any model depicting a time series, the design matrix will have a slight modification: instead of n observations, we have T time points indexed by t, where larger t denotes observations that happen at later timepoints.

$$\boldsymbol{X} = \begin{bmatrix} X_{11} & \dots & X_{1j} & \dots & X_{1p} \\ \vdots & & \vdots & & \vdots \\ X_{t1} & \dots & X_{tj} & \dots & X_{tp} \\ \vdots & & \vdots & & \vdots \\ X_{T1} & \dots & X_{Tj} & \dots & X_{Tp} \end{bmatrix}$$

For the Gaussian Process model in this dissertation, we will use i as the variable for indexing, although much of the principles behind the notation of a time series still apply, i.e. $n = T$ and larger values of i correspond to observations that happen at later timepoints.

2.8 Linear Regression

The fact that we can so easily define a linear combination of Gaussian random variables as a Gaussian random variable lies at the heart of the theory of linear regression. *Linear Regression* is a model determining the strength of the relationship between a dependent variable Y as a linear combination of other independent variables $X_1, \cdots, X_p$.

$$\begin{aligned} \boldsymbol{Y} &= c_1 X_1 + c_2 X_2 + \cdots + c_p X_p \\ &= \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p \\ &= \boldsymbol{\beta}^\top \boldsymbol{X} \text{ (linear regression)} \end{aligned}$$

By changing the weights c_i in our linear combination to β_i, we have our familiar expression of linear regression, where $\boldsymbol{X}$ is our design matrix and $\boldsymbol{\beta}$ is a vector of coefficients:

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix}$$

The aim of linear regression is to *fit*, or to *estimate*, our coefficients $\boldsymbol{\beta}$ appropriately such that a loss function is minimized. A *loss function*, also known as a *cost function*, typically describes a difference between the fitted values as predicted from our model and the true values of the observations.

The most common loss function used in linear regression is the L_2 loss function, which is the square of the difference between estimated and observed values.

$$L_2 = \sum_{i=1}^{n} (Y_i - \boldsymbol{\beta}^\top \boldsymbol{X}_i)^2$$

Note that, in the above equation, $\boldsymbol{X}$ is the design matrix such that $\boldsymbol{X}_i$ is the i-th observation. The $\boldsymbol{\beta}$-coefficients that minimize the loss function are then our *fitted* values, or *trained* parameters. These fitted values are denoted using a hat symbol, $\hat{\boldsymbol{\beta}}$.

$$\hat{\boldsymbol{\beta}} = \arg\max_{\boldsymbol{\beta}} \sum_{i=1}^{n} \left(Y_i - \boldsymbol{\beta}^\top \boldsymbol{X}_i \right)^2$$

In real life, we cannot express $\boldsymbol{Y}$ perfectly as a linear combination of a predetermined set of features $X_1, \cdots, X_p$. Fortunately, because a linear combination of Gaussian variables is a Gaussian variable, we can fit an additional Gaussian variable $X_0 \sim \mathcal{N}(\mu_0, \sigma_0^2)$ to

describe the difference between $\boldsymbol{Y}$ and our most optimal linear combination of $X_1, \cdots, X_p$:

$$X_0 = \boldsymbol{Y} - \boldsymbol{\beta}^\top \boldsymbol{X}$$

This X_0 variable is commonly called noise. It will always exist, because in real life, it is impossible to express $\boldsymbol{Y}$ perfectly as a linear combination of a predetermined set of features $(X_1, \cdots, X_p)$ unless the relationship between $\boldsymbol{Y}$ and $(X_1, \cdots, X_p)$ is deterministic. Introducting an additional variable $X_0 \sim \mathcal{N}(\mu_0, \sigma_0^2)$ is the best way we can continue to describe $\boldsymbol{Y}$ perfectly as a linear combination of Gaussian variables:

$$\boldsymbol{Y} = X_0 + \boldsymbol{\beta}^\top \boldsymbol{X}$$

X_0 is typically incorporated in our design matrix $\boldsymbol{X}$ as a column vector of 1's because the coefficient of X_0 under this linear combination representation of $\boldsymbol{Y}$ is 1. The column vector of 1's is also what mathematicians and statisticians refer to as the intercept term. Thus, to represent this new design matrix in its entirety, we write the following:

$$\boldsymbol{X} = \begin{bmatrix} 1 & X_{11} & \dots & X_{1j} & \dots & X_{1p} \\ \vdots & \vdots & & \vdots & & \vdots \\ 1 & X_{i1} & \dots & X_{ij} & \dots & X_{ip} \\ \vdots & \vdots & & \vdots & & \vdots \\ 1 & X_{n1} & \dots & X_{nj} & \dots & X_{np} \end{bmatrix}$$

Typically, to center X_0 so that the resulting distribution is $\mathcal{N}(0, \sigma_0^2)$, we introduce an additional beta coefficient set to the mean of X_0. Explicitly, we write $\hat{\beta}_0 = \hat{\mu}_0$ because we recall $X_0 - \mu_0 \sim \mathcal{N}(0, \sigma_0^2)$. We can then express $\boldsymbol{Y}$ now as the following linear combination:

$$\boldsymbol{Y} = \beta_0 + \boldsymbol{\beta}^\top \boldsymbol{X} + \mathcal{N}\left(0, \sigma_0^2\right)$$

Finally, as a notational convention, statisticians like to employ ε to denote this zero-centered noise variable $\varepsilon \sim \mathcal{N}\left(0, \sigma_0^2\right)$. It is extremely common to see linear regression expressed in this final manner:

$$\boldsymbol{Y} = \beta_0 + \boldsymbol{\beta}^\top \boldsymbol{X} + \varepsilon$$

2.9 Gaussian Process Regression

Gaussian Processes Regression is an extension of linear regression. There are two common ways of viewing Gaussian Processes. The extension of linear regression is one

view of Gaussian Processes, known as the *weight-space view*. Another view of Gaussian Processes is the *function-space view*, and both will be examined here. From here, we begin using the language of inputs interchangably with dependent variables $\boldsymbol{X}$ and targets interchangably with indepedent variable $\boldsymbol{Y}$.

Recall that a major goal of *inference* is to use observed data to describe a ground truth. Our Gaussian Process Regression model is what we perceive to be the ground truth. Therefore, to conduct inference, there must be ways to describe how likely our observed data fits in our model of ground truth.

2.10 Likelihood Function

A *likelihood function* describes the probability of observing the data conditioned on the parameters of the model we've specified, and is commonly used to describe how likely our observed data fits our model of ground truth. Higher values produced by the likelihood function indicate greater likelihood that the data we observed was generated by the model with the parameters we've specified.

Explicitly, we can write the likelihood function as a function of our data y and parameters θ in any of the following ways:

$$L(\theta) = P(y|\theta) = f_\theta(y) = P(y;\theta)$$

When writing out the likelihood function of the Gaussian distribution, we observe that $\theta = \left(\mu, \sigma^2\right)$, so that the θ in our likelihood is actually a vector of two parameters μ and σ, and we can therefore plug in our values to obtain our likelihood function for one observed data point as follows:

$$\begin{aligned} L(\theta) &= P(y|\theta) \\ &= P(y|\mu, \sigma^2) \\ &= \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2} \end{aligned}$$

We observe that the likelihood function is similar to the probability density function (PDF)!

However, there is a subtlety here: the probability density function describes the probability of obtaining a specific instance of *data from known parameters*, while the likelihood function describes the likelihood of obtaining a specific instance of *parameters*

from observed data. In other words, probability and inference are two sides of a coin – to go from model to data, probability is applied, and to go from data to model, inference is applied.

Moreover, the likelihood function is almost always taken with respect to many observations, while a probability is usually taken with respect to one observation. Thus, likelihood functions are a product of n PDFs, where n is the number of observations.

$$\begin{aligned} L(\theta) &= \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{y_1-\mu}{\sigma}\right)^2} \times \cdots \times \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{y_n-\mu}{\sigma}\right)^2} \\ &= \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{y_i-\mu}{\sigma}\right)^2} \end{aligned}$$

The likelilhood function is powerful in capturing a direct relationship between the observed data and the model parameters. We will use the likelihood function extensively in our treatment of Gaussian Processes under both the weight-space view and the function-space view.

2.11 Weight-Space View

Recall in linear regression, we can express $\boldsymbol{Y} = \hat{\beta}_0 + \boldsymbol{\beta}^\top \boldsymbol{X} + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. The weight-space view treats each Y_i at every i-th time point as a linear regression of $Y_i = \beta_0 + \boldsymbol{\beta}^\top \phi(\boldsymbol{X}_{\leq i}) + \varepsilon$ where $\phi(\boldsymbol{X}_{\leq i})$ is a *basis function* of a subset of the decision matrix of all values up until the i-th time point. A basis function transforms our random variables so that they cover a different basis space. However, regardless of what basis functions we construct, or whether we even decide to construct a basis function at all, we observe the weights $\boldsymbol{\beta}$ are still used in the model to describe Y_i, thus giving this particular formulation the name weight-space view.

We are interested in making inferences about the relationship between inputs and targets, i.e., the conditional distribution of the targets given the inputs, which is known as a discriminative model[9]. This is a less difficult problem than modeling the input distribution itself, known as a generative model[10], where the difficulty is ascribed to more computations needed for a generative model.

The discriminative model $P(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\beta})$ can be derived using the likelihood function as shown below.

[9] A discriminative model learns a conditional probability $P(Y|X)$

[10] A generative model learns a joint probability $P(Y \text{ and } X)$

$$
\begin{aligned}
P(\boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{\beta}) &= \prod_{i=1}^{n} P\left(\boldsymbol{Y}_i \mid \boldsymbol{X}_i, \boldsymbol{\beta}\right) \\
&= \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\left(\boldsymbol{Y}_i - \boldsymbol{\beta}^\top \boldsymbol{X}_i\right)^2}{2\sigma^2}\right) \\
&= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}\left|\boldsymbol{Y} - \boldsymbol{\beta}^\top \boldsymbol{X}\right|^2\right)
\end{aligned}
$$

From the likelihood function, we can conclude that the discriminative model shares the PDF with a Gaussian distribution of mean $\boldsymbol{\beta}^\top \boldsymbol{X}$ and variance $\sigma^2 I$.

$$
P(\boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{\beta}) \sim \mathcal{N}\left(\boldsymbol{\beta}^\top \boldsymbol{X}, \sigma^2 I\right)
$$

2.12 Function-Space View

The function-space view extends our original definition of a Gaussian Process, which we can recall here:

> A Gaussian process $(X_t)_{t>0}$ is a continuous-time stochastic process with the property that for all $n = 1, 2, \ldots$ and $0 \leq t_1 < \cdots < t_n$, the random variables $X_{t_1}, \ldots, X_{t_n}$ have a multivariate normal distribution.

In our examination of the multivariate normal distribution, we identified the two parameters that characterize the multivariate normal distribution, one being the mean vector $\boldsymbol{\mu}$, and the other being the covariance matrix $\boldsymbol{\Sigma}$. We can define both the mean and the covariance as a function of the inputs $\boldsymbol{X}$, thus giving this particular formulation the name function-space view.

Thus, using a change of notation, a Gaussian Process is a collection of random variables with mean function $\mu(\boldsymbol{X})$ and covariance function, or *kernel*, $K(\boldsymbol{X}, \boldsymbol{X}^\top)$.

$$
\begin{aligned}
\mu(\boldsymbol{X}) &= \mathbb{E}[f(\mathbf{X})] \\
K\left(\mathbf{X}, \mathbf{X}^\top\right) &= \mathbb{E}\left[\left(f(\mathbf{X}) - \mu(\mathbf{X})\right)\left(f\left(\mathbf{X}^\top\right) - \mu\left(\mathbf{X}^\top\right)\right)\right]
\end{aligned}
$$

The discriminative model under weight-space view can then be represented as a Gaussian Process with those functions as parameters:

$$P(\boldsymbol{Y} \mid \boldsymbol{X}) \sim \mathcal{GP}\left(\mu(\mathbf{x}), K\left(\boldsymbol{X}, \boldsymbol{X}^\top\right)\right)$$

Here, we introduce a new notation $K\left(\boldsymbol{X}, \boldsymbol{X}^\top\right)$, sometimes shorthanded with just K. While we have just familiarized ourselves with $K\left(\boldsymbol{X}, \boldsymbol{X}^\top\right)$ being a covariance function, we now refer to $K\left(\boldsymbol{X}, \boldsymbol{X}^\top\right)$ in context of Gaussian Processes more commonly as a kernel. The motivation comes from the ability to reduce ambiguity between the covariance matrix and the covariance function, in addition to the application of some principles of Bayesian inference, because we are interested in modeling the changes of our *covariance matrix* upon new observations of data. By treating the covariance matrix as a parameter, the covariance function can then be used in describing changes of the covariance matrix.

Moreover, the covariance function has traditionally had a fixed definition: $\text{Cov}(X_i, X_j) = E(X_i X_j) - E(X_i)E(X_j)$. In spirit of dispelling the notion that there only exists a fixed definition for the covariance function, the language of kernels is used instead.

2.13 Bayesian Inference

Gaussian Processes have gained popularity because of some elegant properties as applied in Bayesian inference.

Bayesian inference is a particular point of view in inference where parameters are treated as random variables with a *prior* density distribution $\pi(\theta)$ before any data has been collected. The prior density distribution is updated into a *posterior* density distribution $g(\theta|y)$ upon observation of data y.

The *Bayesian* point of view is typically contrasted with the *Frequentist* point of view. The fundamental difference on a high level is that Frequentist inference treats the parameters to be fixed, as opposed to being random. As a consequence, Frequentist inference does not rely on a prior distribution on the parameters.

The interpretations and conclusions of Frequentist inference are weaker than those of Bayesian inference because it is harder to infer how flexibly the parameters of the model change as the data changes. However, by the same token, *Bayesian* inference can offer

too much flexibility such that a particular choice of prior distribution on the parameters could result in a less performant model under limited data. Ultimately, both the Bayesian and Frequentist point of views converge as the data size increases.

Bayesian inference derives its name from Bayes' Rule:

$$P(\theta \mid Y) = \frac{P(Y \mid \theta)P(\theta)}{P(Y)}$$

The prior distribution is $P(\theta)$, the likelihood is $P(Y \mid \theta)$ and the posterior distribution is $P(\theta \mid Y)$. $P(Y)$ is effectively treated as a normalizing constant because we recall that all probabilities must sum to 1 over the entire sample space. Typically, it is more succinct to drop the normalizing constant when focusing on the relationship between the posterior and prior distributions under Bayesian inference:

$$P(\theta|Y) \propto P(Y|\theta)P(\theta)$$

Without any structure on the parameter's prior distribution and its relationship with the data via the likelihood function, it is hard to describe what the posterior distribution will be. Fortunately for the Gaussian Process, the posterior distribution of the parameter will always be the same distribution as the prior distribution of the parameter – both distributions will be Gaussian distributions due to the Normal-Normal conjugacy.

Normal-Normal Conjugacy

Given a Gaussian prior distribution on the mean parameter $\mu \sim \mathcal{N}(\mu_0, \sigma_0^2)$ and observations $\boldsymbol{X} = (x_1, \cdots, x_n)$, the posterior distribution $\mu|\boldsymbol{X}$ is a Gaussian distribution with updated parameters:

$$P(\mu|\boldsymbol{X}) \sim N\left(\left(\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}\right)^{-1}\left(\frac{\mu_0}{\sigma_0^2} + \frac{\sum_{i=1}^n x_i}{\sigma^2}\right), \left(\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}\right)^{-1}\right)$$

This is known as the Normal-Normal conjugacy.

Proof:

Recall that in Bayesian inference, $P(\theta \mid Y) \propto P(Y \mid \theta)P(\theta)$. By identifying our parameter of interest to be $\theta = \mu$, we can substitute the likelihood function $P(\boldsymbol{X} \mid \mu)$

and the prior $P(\mu)$ to solve for the posterior $P(\mu \mid \boldsymbol{X})$.

$$
\begin{aligned}
P(\mu \mid \boldsymbol{X}) &\propto P(\boldsymbol{X}|\mu)P(\mu) \\
&= \left(\prod_{i=1}^{n} \exp\left(-\frac{1}{2\sigma^2}(x_i-\mu)^2\right)\right) \exp\left(-\frac{1}{2\sigma_0^2}\left(\mu-\mu_0\right)^2\right) \\
&= \exp\left(-\sum_{i=1}^{n}\frac{1}{2\sigma^2}(x_i-\mu)^2 - \frac{1}{2\sigma_0^2}\left(\mu-\mu_0\right)^2\right) \\
&\propto \exp\left(-\frac{1}{2}\left(\left(\frac{n}{\sigma^2}+\frac{1}{\sigma_0^2}\right)\mu^2 - 2\left(\frac{\sum_{i=1}^{n} x_i}{\sigma^2}+\frac{\mu_0}{\sigma_0^2}\right)\mu\right)\right) \\
&\propto \exp\left(-\frac{a}{2}\left(\mu-\frac{b}{a}\right)^2\right) \text{ where } a = \left(\frac{n}{\sigma^2}+\frac{1}{\sigma_0^2}\right), b = \left(\frac{\sum_{i=1}^{n} x_i}{\sigma^2}+\frac{\mu_0}{\sigma_0^2}\right)
\end{aligned}
$$

The final expression is proportional to the PDF of a Normal distribution containing all the terms that interact with the parameter μ. Thus, we can conlude that $\mu \mid X$ has the PDF of a Normal distribution with mean parameter b/a and variance parameter $1/a$.

$$\mu \mid \boldsymbol{X} \sim \mathcal{N}\left(\frac{b}{a}, \frac{1}{a}\right)$$

Substituting in our values of $a = \left(\frac{n}{\sigma^2}+\frac{1}{\sigma_0^2}\right)$ and $b = \left(\frac{\sum_{i=1}^{n} x_i}{\sigma^2}+\frac{\mu_0}{\sigma_0^2}\right)$, the posterior distribution is a Normal distribution with updated parameters.

$$\mathcal{N}\left(\left(\frac{1}{\sigma_0^2}+\frac{n}{\sigma^2}\right)^{-1}\left(\frac{\mu_0}{\sigma_0^2}+\frac{\sum_{i=1}^{n} x_i}{\sigma^2}\right), \left(\frac{1}{\sigma_0^2}+\frac{n}{\sigma^2}\right)^{-1}\right)$$

Bayesian Weight-Space View

We can apply Bayesian inference in the weight-space view by imposing a prior distribution on the parameter weights $\boldsymbol{\beta}$ with mean 0 and covariance matrix Σ_p:

$$\boldsymbol{\beta} \sim \mathcal{N}\left(\mathbf{0}, \Sigma_p\right)$$

Using Bayes' rule, there exists a posterior distribution of the parameter weights, which we can derive.

$$
\begin{aligned}
P(\boldsymbol{\beta} \mid \mathbf{y}, X) &= \frac{P(\mathbf{y} \mid X, \boldsymbol{\beta}) P(\boldsymbol{\beta})}{P(\mathbf{y} \mid X)} \\
&= \frac{P(\mathbf{y} \mid X, \boldsymbol{\beta}) P(\boldsymbol{\beta})}{\int P(\mathbf{y} \mid X, \boldsymbol{\beta}) P(\boldsymbol{\beta}) d\boldsymbol{\beta}} \\
&\propto P(\mathbf{y} \mid X, \boldsymbol{\beta}) P(\boldsymbol{\beta}) \\
&\propto \exp\left(-\frac{1}{2\sigma^2} (\mathbf{y} - X\boldsymbol{\beta})^\top (\mathbf{y} - X\boldsymbol{\beta})\right) \exp\left(-\frac{1}{2}\boldsymbol{\beta}^\top \Sigma_p^{-1} \boldsymbol{\beta}\right) \\
&\propto \exp\left(-\frac{1}{2}(\boldsymbol{\beta} - \overline{\boldsymbol{\beta}})^\top \left(\frac{1}{\sigma^2} X^\top X + \Sigma_p^{-1}\right) (\boldsymbol{\beta} - \overline{\boldsymbol{\beta}})\right)
\end{aligned}
$$

where

$$
\overline{\beta} = \sigma^{-2} \left(\sigma^{-2} X^\top X + \Sigma_p^{-1}\right)^{-1} X^\top \mathbf{y}
$$

This result implies

$$
p(\boldsymbol{\beta} \mid X, \mathbf{y}) \sim \mathcal{N}\left(\overline{\boldsymbol{\beta}} = \frac{1}{\sigma^2} A^{-1} X^\top \mathbf{y}, A^{-1}\right)
$$

where

$$
A = \sigma^{-2} X^\top X + \Sigma_p^{-1}
$$

Finally, we can use the posterior distribution to obtain the predictive distribution, which averages all linear models with respect to the posterior. We can write $\mathbf{x}_*$ as a new input for f_*, so that f_* is the new generative model under newly observed $\mathbf{x}_*$. This procedure of finding the updated generative model is also known as finding the marginal distribution, or *marginalization*.

$$
\begin{aligned}
P\left(f_* \mid \mathbf{x}_*, X, \mathbf{y}\right) &= \int P\left(f_* \mid \mathbf{x}_*, \boldsymbol{\beta}\right) P(\boldsymbol{\beta} \mid X, \mathbf{y}) d\boldsymbol{\beta} \\
&= \int \mathbf{x}_* \boldsymbol{\beta} P(\boldsymbol{\beta} \mid X, \mathbf{y}) d\boldsymbol{\beta} \\
&= \mathcal{N}\left(\frac{1}{\sigma_0^2} \mathbf{x}_* A^{-1} X \mathbf{y}, \mathbf{x}_*^\top A^{-1} \mathbf{x}_*\right)
\end{aligned}
$$

Ridge Regression

From the weight-space view, we obtain an interesting result, which is that our choice of priors corresponds directly to a constraint function on our weights β.

Recall that in linear regression, we are finding an optimal β which minimizes the loss function:

$$L_2 = \sum_{i=1}^{n} \left(Y_i - \boldsymbol{\beta}^\top \boldsymbol{X}_i \right)^2$$

$$\hat{\boldsymbol{\beta}} = \arg\max_{\boldsymbol{\beta}} \sum_{i=1}^{n} \left(Y_i - \boldsymbol{\beta}^\top \boldsymbol{X}_i \right)^2$$

To find the optimal $\hat{\boldsymbol{\beta}}$, we set the derivative of our loss function to 0, observing a local minimum because the second derivative of our loss function at that point is positive.

$$\hat{\boldsymbol{\beta}} = \left(X^\top X \right)^{-1} X^\top Y$$

In ridge regression, an additional penalty term is added to the loss function:

$$L_{ridge} = \sum_{i=1}^{n} \left(Y_i - \boldsymbol{\beta}^\top \boldsymbol{X}_i \right)^2 + \lambda \sum_{j=1}^{n} \beta_j^2$$

$$\hat{\boldsymbol{\beta}} = \arg\max_{\boldsymbol{\beta}} \sum_{i=1}^{n} \left(Y_i - \boldsymbol{\beta}^\top \boldsymbol{X}_i \right)^2 \text{ subject to } \lambda \sum_{i=1}^{n} \beta_j^2 \leq 1$$

Just as in the linear regression setting, to find the optimal $\hat{\boldsymbol{\beta}}$ in ridge regression, we set the derivative of our loss function to 0, observing a local minimum because the second derivative of our loss function at that point is positive. The only difference is that the loss function is now L_{ridge} as opposed to L_2.

$$\widehat{\beta}_{\text{ridge}} = \left(X^\top X + \lambda I_p \right)^{-1} X^\top y$$

Our posterior distribution in the weight-space view contains a posterior mean $\overline{\beta}$ that is extremely similar to $\widehat{\beta}_{\text{ridge}}$.

$$p(\boldsymbol{\beta} \mid X, \mathbf{y}) \sim \mathcal{N}\left(\overline{\boldsymbol{\beta}} = \frac{1}{\sigma^2} A^{-1} X^\top \mathbf{y}, A^{-1} \right)$$

where $A = \sigma^{-2} X^\top X + \Sigma_p^{-1}$.

$$\bar{\beta} = \left(X^\top X + \frac{1}{\sigma^2} \Sigma_p^{-1} \right)^{-1} X^\top y$$

We can pattern-match our terms to notice that $\frac{1}{\sigma^2}\Sigma_p^{-1}$ inside $\bar{\beta}$ is analogous to the λI_p in $\widehat{\beta}_{\text{ridge}}$, thereby relating ridge regression and the Gaussian prior distribution in an elegant manner.

Bayesian Function-Space View

Recall in the function-space view, we formulate a Gaussian Process to be

$$P(\boldsymbol{Y} \mid \boldsymbol{X}) \sim \mathcal{GP}\left(\mu(\mathbf{X}), K\left(\boldsymbol{X}, \boldsymbol{X}^\top\right)\right)$$

Our parameters under this representation are the mean parameter $\mu(\boldsymbol{X})$ and the covariance function, also known as the kernel, $K\left(\boldsymbol{X}, \boldsymbol{X}^\top\right)$. Because parameters are treated as unknown variables under Bayesian inference, we can place a prior on both the mean function and kernel directly under this function-space view.

Typically, a good choice of a prior density function is $\mu(\mathbf{X}) = 0$, since we'd like to center our Gaussian process at 0 for easier mathematical representation and computation. Then, we would update the mean accordingly from the data via the likelihood function.

The kernel, however, is more interesting to study. We can define our kernel to be any function, not limited to the definition of the traditional covariance function. The only requirement for a valid kernel is that the covariance matrix produced by the kernel using the observations as inputs is *positive definite*.

For a covariance matrix to be *positive semi-definite*, the covariance matrix a symmetric matrix with all positive eigenvalues. It is mathematically tedious to check the matrix with all positive eigenvalues, but fortunately, there exists efficient computation and decomposition techniques that we can employ to study positive definiteness of matrices. These techniques will be examined in the computation chapter.

The process of defining a new valid kernel from scratch is not always trivial, because there must be a compelling assumption to justify pre-defined kernels in modeling a Gaussian Process. The choice of kernels is a question frequently encountered in hyperparameter tuning, where computation is applied to find optimal parameters that cannot be learned from *fitting* or *training* the data. *Hyperparameters* are pre-defined throughout the model-fitting process, and therefore can be seen as an extension of our assumptions rather than as fitted values from inference or learning.

Because the choice of kernels requires certain assumptions about the data, in the case where there are no particularly strong views on how the data is to behave, a popular

choice of kernel is the *squared exponential kernel*, also known as the radial basis function kernel, because it can be shown that the squared exponential kernel, corresponds to the defined covariance function $\text{Cov}(X_i, X_j) = E(X_i X_j) - E(X_i)E(X_j)$ for a Gaussian Process.

$$\begin{aligned} K\left(\mathbf{x}_p, \mathbf{x}_q\right) &= \exp\left(-\frac{1}{2}\left|\mathbf{x}_p - \mathbf{x}_q\right|^2\right) \\ &= E(f(\mathbf{x}_p)f(\mathbf{x}_q)) - E(f(\mathbf{x}_p))E(f(\mathbf{x}_q)) \\ &= \text{Cov}\left(f\left(\mathbf{x}_p\right), f\left(\mathbf{x}_q\right)\right) \end{aligned}$$

The squared exponential kernel is an example of a *stationary* kernel, which is any function containing $r = |\boldsymbol{x} - \boldsymbol{x}^\top|$. If the kernel only interacts with $\boldsymbol{x}$ and $\boldsymbol{x}^\top$ via the expression $r = |\boldsymbol{x} - \boldsymbol{x}^\top|$, as is the case with the squared exponential kernel, then we say that the kernel is *isotropic*.

The kernel provides the underlying structure to any Gaussian Process model. Therefore, any formulation of a Gaussian Process demands an assumption about the kernel.

3

Binary Outcomes

In traditional Gaussian Process models, the data is assumed to follow a Gaussian distribution, which possesses many elegant properties. By assuming the prior distribution of the data is a Gaussian distribution, the predictive posterior distribution of the data is also a Gaussian distribution.

However, not all real life phenomenon behave according to a Gaussian distribution. Some phenomenon take on values that are discrete, rather than values that are continuous. For such phenomenon, it is a poor assumption to assume that the data follows a Gaussian distribution because the Gaussian distribution takes on continuous values.

Instead, a different distribution must be used in describing the data, one that follows a more appropriate assumption of the discrete nature of the data.

While a different distribution is needed to describe data that is discreet, it need not be the case that the Gaussian Process model fails entirely as a model for the data. Rather, a Gaussian Process model could be useful in fitting and modeling the parameters of the newly defined distribution.

This chapter is dedicated to examining Gaussian Processes on such non-Gaussian data, specifically addressing binary outcomes data. Binary outcomes are perhaps the simplest examples of discreet data, and from this simplicity comes ubiquity. Binary data is easy to define, collect, and store, and is used extensively in product testing and A/B testing, providing a real-world motivation for understanding Gaussian Processes on binary outcomes.

3.1 Classification

The difference between discrete and continuous data can be better formalized into two classes of problems – classification and regression. By definition, a discrete value is any value that takes on integer values, so we write $D \in \mathbb{Z}$ if the data D is discrete, while continuous data are values that can take on any value in the real number line, so we write $D \in \mathbb{R}$ if the data D is continuous.

Classification problems must assign a discrete target $\boldsymbol{Y} \in \mathbb{Z}$ based on some given inputs $\boldsymbol{X}$.

Regression problems must assign a continuous target $\boldsymbol{Y} \in \mathbb{R}$ based on some given inputs $\boldsymbol{X}$.

From the definitions provided, the distinction between *classification* and *regression* problems is decided completely by the nature of values we are trying to assign, or to *predict*, whether $\boldsymbol{Y} \in \mathbb{Z}$ or $\boldsymbol{Y} \in \mathbb{R}$. Note that there is no restriction on the behavior of the inputs $\boldsymbol{X}$ in both cases.

3.2 Binary Target

A discrete target becomes binary if the target only has two classes in the classification scheme, which we can denote $\{C_1, C_2\}$.

Target synonymous with the dependent variable as well as the response variable. Because of all these synonyms, it's often clearest to reference the mathematical variable Y as defined to be all of these things to lessen confusion over language.

The two classes can be translated into ordinal numbers, indicated by 0 for the first class and 1 for the second class. Traditionally, the class indicated 0 is the default outcome (C_1), and the class indicated 1 is the non-default outcome (C_2). Thus for any binary outcome, there exists a function f which maps classes into 0 and 1.

$$f(C) \mapsto \begin{cases} 0 & \text{if } C = C_1 \\ 1 & \text{if } C = C_2 \end{cases}$$

We can use this function f to ultimately define Y, the target, so that $f(C) = Y$. Then for each Y_i, an assignment is made for whether the i-th observation was in the first class

C_1 or the second class C_2.

$$Y_i \mapsto \begin{cases} 0 & \text{if } C_i = C_1 \\ 1 & \text{if } C_i = C_2 \end{cases}$$

Examples

Here are some examples to make classification more explicit.

1. A data scientist is interested in testing whether a new placement of banner ads will cause users to visit a certain website. The classification is C_1 being defined as no users are visiting the website and C_2 being defined as at least one user is visiting the website.
2. A quantitative researcher is interested in seeing whether a trade will cause the market price of a stock to go up or down by the end of the day. The classification is C_1 being defined as the market price going up end of day and C_2 being defined as the market price going down end of day.
3. A sabermetrician is interested in whether there is a home-team advantage for baseball teams. The classification is C_1 being defined as the home team winning and C_2 being defined as the home team losing.
4. A biostatistician is interested in whether a new quantum drug causes a cat to stay alive under radioactive poison. The classification is C_1 being defined as the cat dies with the quantum drug and C_2 being defined as the cat stays alive with the quantum drug.

Note that in each of the cases, a clear delineation is made between the two classes. All outcomes can either be classified into C_1 or C_2 but not both. Of course, some assumptions are made such that C_1 and C_2 together define the entire sample space of outcomes.

Example 1 with the data scientist assumes that there cannot exist a negative number of users visiting the website. Example 2 with the quantitative researcher assumes the market price cannot stay the same end of day (intriguingly, this assumption can be verified using the Black-Scholes model, which is an application of Gaussian Processes on stock movement, which, by nature of there being a continuous distribution, places zero-mass probability on stock movement staying at 0). Similarly, Example 3 with the

sabermetrician assumes that there can be no ties (which is true in baseball, but not in soccer!). Finally, although Example 4 sounds eerily similar to the Schrödinger's cat thought experiment, where a cat can be simultanously alive and dead, in order for the classes to be binary, the assumption is ultimately made that the cat can either be dead or alive with the quantum drug, but not both.

3.3 Binary Treatment Effects

In the preceding examples, we identified the binary classes that the target would be classified into.

Binary treatment effects, or *binary actions*, also follow a similar definition in that actions themselves are discreet and binary, taking on the value 0 if action was not taken and the value 1 if action was taken. However, a critical difference between binary actions and binary outcomes is that binary outcomes is always used to describe the target $\boldsymbol{Y}$ while binary action exists as a feature or input in our design matrix $\boldsymbol{X}$. We use the notation A_i as a scalar to denote the i-th binary action, which takes on values of 0 or 1 based on whether action was taken.

$$A_i \mapsto \begin{cases} 0 & \text{if action } A_i \text{ not taken} \\ 1 & \text{if action } A_i \text{ taken} \end{cases}$$

Because $\boldsymbol{A}$ is an input in our design matrix, the dimensions of vector $\boldsymbol{A}$ must be of the same size as the number of rows in the design matrix $\boldsymbol{X}$. Thus if $\boldsymbol{X}$ has n observations, the number of rows is n and $\boldsymbol{A}$ must be a column vector of dimension n.

$$\boldsymbol{A} = \begin{bmatrix} A_0 \\ \vdots \\ A_i \\ \vdots \\ A_n \end{bmatrix}$$

Binary actions are important in causal inference, a branch of inference which determines causality between input $\boldsymbol{A}$ and outcomes $\boldsymbol{Y}$.

3.4 Causal Inference

In typical studies, we are passively observing a relationship between inputs $\boldsymbol{X}$ and target $\boldsymbol{Y}$, so that any significant relationship discovered between $\boldsymbol{X}$ and $\boldsymbol{Y}$ can best be characterized as a strong correlation. The study of causal relationships between $\boldsymbol{X}$ and $\boldsymbol{Y}$ is causal inference.

In order to make the statement that $\boldsymbol{X}$ causes $\boldsymbol{Y}$, we employ two popular frameworks in causal inference – counterfactual random variables or directed acyclic graphs.

3.5 Counterfactual Model

The counterfactual model attempts to model counterfactual outcomes, that is, the outcomes that would have happened had the opposite action been applied.

Continuing with our notation as before, we let A be a binary treatment variable where $A = 1$ means "action taken" and $A = 0$ means "action not taken."

$$A_i \mapsto \begin{cases} 0 & \text{if action } A_i \text{ not taken} \\ 1 & \text{if action } A_i \text{ taken} \end{cases}$$

Here is where the subtlety arrises – the classes in the counterfactual model themselves are no longer the values that target $\boldsymbol{Y}$ can take on, but a conditional variable on binary actions $\boldsymbol{A}$. The classes defined here to be *potential outcomes*, are the conditional variable $\boldsymbol{Y}|\boldsymbol{A}$, corresponding to the two outcomes of $\boldsymbol{Y}$ when action $\boldsymbol{A}$ is applied and $\boldsymbol{Y}$ when action $\boldsymbol{A}$ is not applied. One will always be observed and the other not observed, but both exist under the counterfactual model.

$$Y \mapsto \begin{cases} C_0 & \text{if } A = 0 \\ C_1 & \text{if } A = 1 \end{cases}$$

A more succinct representation of Y with respect to C and A is the following:

$$Y = C_A$$

Note that in any counterfactual model, Y need not be binary. $\boldsymbol{Y}$ can take on continuous values, since it is not the values of Y that are binary, but the actions A that must be binary in order for there to exist binary *potential outcomes*.

From here, we observe that when $A = 0$ we don't observe C_1, in which case we say that C_1 is a counterfactual since it is the outcome that would have happened had the opposite action been applied $(A = 1)$.

Similarly, when $A = 1$ we don't observe C_0, and we say that C_0 is counterfactual.

The *average causal effect*, or average treatment effect, is then calculated to be the following. We use the parameter θ to mean the difference between all potential outcomes where action was applied (C_1) and the outcomes where action was not applied C_0

$$\theta = \mathbb{E}\left(C_1\right) - \mathbb{E}\left(C_0\right)$$

If Y was a binary outcome, however, it may not make as much sense to use the average treatment effect since Y can only take on values of 0 or 1. A more appropriate parameter may be the causal odds ratio, which measures the ratio of the causal odds $\frac{P(Y=1)}{P(Y=0)}$ when action was applied to when action was not applied.

$$\theta = \frac{\mathbb{P}\left(C_1 = 1\right)}{\mathbb{P}\left(C_1 = 0\right)} \div \frac{\mathbb{P}\left(C_0 = 1\right)}{\mathbb{P}\left(C_0 = 0\right)}$$

Causal relative risk is an additional parameter one may wish to use, which focuses only on the ratio of $Y = 1$ when action was applied to when action was not applied.

$$\theta = \frac{\mathbb{P}\left(C_1 = 1\right)}{\mathbb{P}\left(C_0 = 1\right)}$$

Finally, to illustrate the point that causation is not association, we consider what it means to find association (α) between Y and A, where we defined $I(A_i = 1)$ to be an indicator random variable where the i-th observation contains action applied.

$$\begin{aligned}
\alpha &= \mathbb{E}(Y \mid A = 1) - \mathbb{E}(Y \mid A = 0) \\
&= \frac{1}{\sum_{i=1}^{n} I(A_i = 1)} \sum_{i=1}^{n} Y_i \times I(A_i = 1) \\
&- \frac{1}{\sum_{i=1}^{n} I(A_i = 0)} \sum_{i=1}^{n} Y_i \times I(A_i = 0)
\end{aligned}$$

Association only tells half of the story, because $Y|A = 1$ contain only the observed cases of Y where action has been applied as opposed to all possible outcomes where action has been applied. Similarly, $Y|A = 0$ contains only observed cases of Y where action has not been applied, not capturing all potential outcomes. The treatment effect does capture all

potential outcomes, and therefore, provides a measure of true causality.

$$\begin{aligned}\theta &= \mathbb{E}\left(C_1\right) - \mathbb{E}\left(C_0\right) \\ &= \frac{1}{n}\sum_{i=1}^{n} C_{1i} \\ &- \frac{1}{n}\sum_{i=1}^{n} C_{0i}\end{aligned}$$

Randomization

While we can never actualize both potential outcomes C_0 and C_1, we can use randomization to identify a causal effect.

This is because random assignment creates independence between between the action performed A and potential outcomes C_0, C_1, causing our treatment effect to be equivalent to the asociation.

$$\begin{aligned}\theta &= \mathbb{E}\left(C_1\right) - \mathbb{E}\left(C_0\right) \\ &= \mathbb{E}\left(C_1 \mid A = 1\right) - \mathbb{E}\left(C_0 \mid A = 0\right) \text{ since } X \text{ is independent to } \left(C_0, C_1\right) \\ &= \mathbb{E}(Y \mid A = 1) - \mathbb{E}(Y \mid A = 0) \text{ since } Y = C_A \\ &= \alpha\end{aligned}$$

The association α becomes a consistent estimator of the average causal effect θ. Recall that a statistic is *consistent* if it converges to the parameter in probability as the number of observations increase.

$$\begin{aligned}\widehat{\theta} &= \hat{\alpha} \\ &= \mathbb{E}(Y \mid A = 1) - \mathbb{E}(Y \mid a = 0) \\ &= \frac{1}{n_1}\sum_{i=1}^{n} Y_i A_i - \frac{1}{n_0}\sum_{i=1}^{n} Y_i\left(1 - A_i\right)\end{aligned}$$

where $n_1 = \sum_{i=1}^{n} A_i$ and $n_0 = \sum_{i=1}^{n}\left(1 - A_i\right)$.

Randomization is considered a key tenet to any scientific study trying to identify causality between variables.

3.6 Directed Acyclic Graphs

The idea behind causal inference can also be formalized in directed acyclic graphs (DAG).

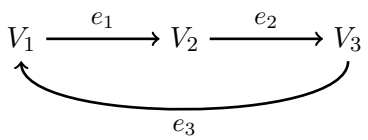
A *directed graph* $G = (V, E)$ consists of a set of nodes or *vertices* V and edges E. In a directed graph, each edge E points from a source node u to a destination node v, and we can write $e = (u, v)$.

A *path* on a graph is a sequence of nodes $v_1, v_2, \cdots, v_k$ such that the edge $(v_i, v_{i+1}) \in E$ for $i = 1, \cdots, k-1$.

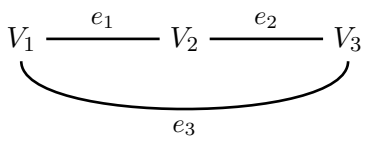
A *cycle* is a path $v_1, v_2, \ldots v_k$ with $k \geq 3$ such that $v_1 = v_k$.

Then, as the name implies, a directed acyclic graph is a directed graph where there are no cycles.

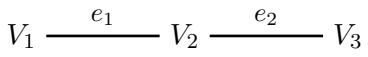
Directed cyclic graph:



Undirected cyclic graph



Undirected acyclic graph



Directed acyclic graph

$$V_1 \xrightarrow{e_1} V_2 \xrightarrow{e_2} V_3$$

If we were to replace the nodes in the directed acyclic graph with random variables, and the directed edges represented causal relationships, we have constructed a pictoral representation of causal inference using directed acyclic graphs. Note that cyclic graphs make no sense in context of causal inference, because there exists a chicken-vs-egg problem since any of the random variables in the cycle would exist simultaneously as a causal variable as well as a response variable.

$$X \xrightarrow{X \text{ causes } Y} Y \xrightarrow{Y \text{ causes } Z} Z$$

Bayesian Networks

Directed acyclic graphs provide a rich illustration of independence relations between variables. This representation is frequently described as a *Bayesian network*, even though there is nothing particular Bayesian about this representation, because we recall that Bayesian inference places a prior distribution on the parameters, and the graphs here make no assumption about prior distributions of variables. We will stick to the language of DAG over Bayesian networks because of its cleaner definition and connotation.

It is useful to define the specific types of relationships between random variables under a DAG.

Parent

Random variable X is a *parent* of random variable Y if there is an edge from X to Y. Equivalently, this also implies X causes Y.

$$X \longrightarrow Y$$

Child

Random variable X is a *child* of random variable Y if there is an edge from Y to X. Equivalently, this also implies Y causes X.

$$Y \longrightarrow X$$

Ancestor

Random variable X is an *ancestor* of random variable Y if there is a path from X to Y. This also implies X causes Y.

$$X \longrightarrow \cdots \longrightarrow Y$$

Descendent

Random variable X is a *descendent* of random variable Y if there is a path from Y to X. This also implies Y causes X.

$$Y \longrightarrow \cdots \longrightarrow X$$

Collider

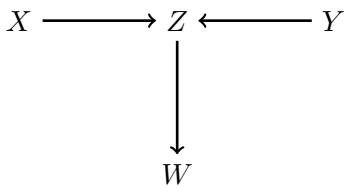
A configuration between three variables X, Y, Z is a *collider* if there exists an edge pointing from X to Y and an edge pointing from Z to Y. We say there exists a collider at Y. Note that the collider relationship must apply to three variables, as opposed to two.

$$X \longrightarrow Y \longleftarrow Z$$

3.7 Probability and DAG

For any directed acyclic graph, we can represent a joint probability as a product of marginal probabilities and conditional probabilities.

We can make this relationship between probability and DAG more explicit in the following example.



Using our definitions in the previous section, we observe several relationships:

1. X is a parent of Z, which also implies Z is a child of X
2. Y is a parent of Z, which also implies Z is a child of Y
3. Z is a parent of W, which also implies W is a child of Z
4. X is an ancestor of W, which also implies W is a descendent of X
5. Y is an ancestor of W, which also implies W is a descendent of Y
6. X, Z, Y form a collider at Z

When calculating the joint probability $f(x, y, z, w)$, we want to start at the source nodes in our DAG, in other words, nodes with no parents, because these nodes are not dependent on any other node. In our example, this would correspond to nodes X and Y with marginal probabilities $f(x)$ and $f(y)$ respectively.

Afterwards, we traverse our DAG by going along the parents to the children via the edges. In doing so, we arrive at some new nodes, and we must multiply our starting marginal probabilities now with conditional probabilities. The conditional probabilities are conditioned on all the nodes that are direct parents. When placed in context of our

example, the next node as we traverse from both X and Y would be Z. Z's direct parents are X and Y, so we write $f(z|x,y)$ as the conditional probability.

Next, from Z, would traverse to its child node, which is W. We observe that W has a direct parent which is Z, so the conditional probability of W is $f(w|z)$.

Seeing that there are no more children nodes left, we stop and multiply all the probabilites we've stores in our traversal to find the joint probability. This would correspond to

$$f(x,y,z,w) = f(x)f(y)f(z|x,y)f(w|z)$$

Algorithm

One might notice that the description of our procedure was quite verbose. Indeed, we can provide a more succinct summary of our procedure via *algorithms*.

Algorithms simplify procedures into units of computation and units of storage. Algorithms provides two benefits in that it simplifies the English for us in addition to describing procedures that computers can understand, since computers operate on systems of computation and systems of storage.

Algorithm 1: DAG to Probability

Initialize $q := []$ (q is a queue of nodes)
Initialize $f := []$ (f contains probabilites for DAG)
Loop $\forall s \in V$ where $\nexists s'$ such that $(s', s) \in E$:
  $q.push(s)$
Loop while not $q.empty$:
  $s = q.pop()$ (pops first element f)
  if $\nexists s'$ such that $(s', s) \in E$:
    $f.push(f(s))$
  else:
    Initialize $c := []$ (c contains conditional variables)
    Loop $\forall s' \in V$ where $(s', s) \in E$:
      $c.push(s')$
    $f.push(f(s|c))$
Return f

Using Directed Acyclic Graphs, we can break down the joint probability $f(x,y,z,w)$ into marginal and conditional probabilities, which provides a useful technique in relating the generative model (characterized by the joint probability) and discriminative model (characterized by marginal and conditional probabilities).

3.8 Independence

We recall that for two random variables X, Y to be independent,
$P(X \text{ and } Y) = P(X)P(Y)$. Then, using Bayes' rule, we can reorder this expression to get

$$P(X|Y) = \frac{P(X \text{ and } Y)}{P(Y)} = \frac{P(X)P(Y)}{P(Y)} = P(X)$$

Independence can be made more intuitive using the language of conditioning: If $P(X|Y) = P(X)$, then the probabilistic behavior of X does not change whether new information about Y is presented.

We can use DAG to identify independence between variables by observing that every variable is independent of its ancestors conditioned on the parents. This is also known as the *Markov condition*, where every variable is independent of its history conditioned on the previous step.

However, to generalize indepedence properties further, we can use "d-separation" rules, or "directed separation."

3.9 D-Separation

D-separation is a property characterizing the relationship between two variables with respect to a set of variables. X and Y are *d-separated* given a set of vertices Z if there exists no *undirected* path P between X and Y such that

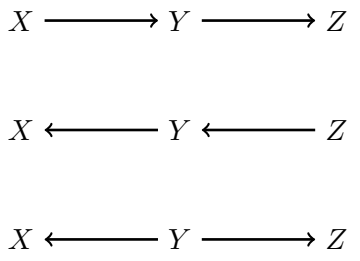
1. Every collider in P has a descendent in Z
2. No other vertex in P is in Z

If X and Y are not d-seperated, they are d-connected.

The formalization of d-separation can be captured completely by three explicit rules, with examples provided.

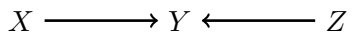
First Rule of d-separation

If X and Z are d-connected and Y is not a collider, X and Z are d-separated given Y.



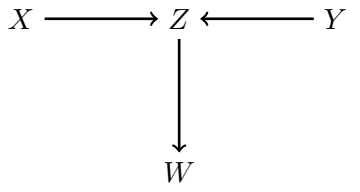
Second Rule of d-separation

If X and Z collide at Y, then X and Z are d-separated, but they are d-connected given Y.



Third Rule of d-separation

Conditioning on the descendant of a collider has the same effect as conditioning on the collider. X and Z are d-separated but they are d-connected given W.



3.10 Binary Outcomes

Having defined directed acyclic graphs, we can now formally define binary outcomes.

In a binary outcome, the target $\boldsymbol{Y}$ is a binary. Recall our definition of binary targets:

$$Y_i \mapsto \begin{cases} 0 & \text{if } C_i = C_1 \\ 1 & \text{if } C_i = C_2 \end{cases}$$

Moreover, $\boldsymbol{Y}$ is a column vector of $n \times 1$ indicating n observations.

$$\boldsymbol{Y} = \begin{bmatrix} Y_1 \\ \vdots \\ Y_i \\ \vdots \\ Y_n \end{bmatrix}$$

Because we are interested in outcomes, we want to identify causal effects of an action on $\boldsymbol{Y}$. We recycle our notation of A_i, where A_i is an indicator variable for binary action for the i-th observation: 1 if action was applied and 0 if action was not applied.

$$A_i \mapsto \begin{cases} 0 & \text{if action } A_i \text{ not taken} \\ 1 & \text{if action } A_i \text{ taken} \end{cases}$$

Similar to $\boldsymbol{Y}$, we can concatenate values of A_i into a column vector $\boldsymbol{A}$ of dimension $n \times 1$ indicating n observations.

$$\boldsymbol{A} = \begin{bmatrix} A_1 \\ \vdots \\ A_i \\ \vdots \\ A_n \end{bmatrix}$$

Moreover, there often exists other variables that are not A in our design matrix. Let us group these variables as $\boldsymbol{X}$, which is also known as the *matrix effects*.

$$\boldsymbol{X} = \begin{bmatrix} X_{11} & \dots & X_{1j} & \dots & X_{1p} \\ \vdots & & \vdots & & \vdots \\ X_{i1} & \dots & X_{ij} & \dots & X_{ip} \\ \vdots & & \vdots & & \vdots \\ X_{n1} & \dots & X_{nj} & \dots & X_{np} \end{bmatrix}$$

Note that $\boldsymbol{Y}$ is a function of both $\boldsymbol{A}$ and $\boldsymbol{X}$. If this was a linear regression formulation, we could simply express $\boldsymbol{Y}$ as a linear combination of $\boldsymbol{A}$ and $\boldsymbol{X}$:

$$\boldsymbol{Y} = \beta_A^\top \boldsymbol{A} + \beta_X^\top \boldsymbol{X}$$

However, because the targets $\boldsymbol{Y}$ are binary outcomes, wecannot use vanilla linear regression to describe the effects of $\boldsymbol{A}$ and $\boldsymbol{X}$ on $\boldsymbol{Y}$. This is because $Y_i|A_i, X_i$ is generated from a $\text{Bern}(\pi)$ distribution.

Bernoulli Distribution

The Bernoulli distribution fundamentally describes all binary data. The only parameter characterizing the Bernoulli distribution is π, the probability of success.

If $X \sim \text{Bern}(\pi)$, then the probability mass function (PMF) is

$$P(X = k) = p^k(1-p)^{1-k}$$

Note that because X is binary, k can either be 1 (success) or 0 (failure). This implies $P(X = 1) = p$ and $P(X = 0) = 1 - p$ if we were to plug in values of $k = \{1, 0\}$ respectively into the probability mass function.

3.11 Gaussian Process on π

Because our target data Y is a binary outcome and therefore is not Gaussian-distributed, we cannot fit a Gaussian Process on the data Y. We must use a Gaussian Process to fit the parameters of a distribution, as opposed to fitting the observations directly.

The only parameter characterizing binary outcomes is the probability parameter π under the Bernoulli distribution. Therefore, we will use a Gaussian Process on the parameter π.

We can identify the parameter π as a function of our inputs A_i, X_i. This leads us to conclude that $Y_i|A_i, X_i \sim \text{Bern}(\pi(A_i, X_i))$.

For some more intuition from a causal inference perspective, we can consider there to be two parameters π_0 and π_1 that correspond to the true potential outcomes when action is not applied (C_0) and when action is applied (C_1). Then, assuming the actions were randomized, $C_0 \sim \text{Bern}(\pi_0(A_i = 1, X_i))$ and $C_1 \sim \text{Bern}(\pi_1(A_i = 0, X_i))$. Then

$\pi(A_i, X_i)$ is the generalizable parameter which becomes π_0 when $Y_i = C_0$ or becomes π_1 when $Y_i = C_1$.

$$\pi(A_i, X_i) = \begin{cases} \pi_0(A_i = 0, X_i) & \text{if } Y_i = C_0 \\ \pi_1(A_i = 1, X_i) & \text{if } Y_i = C_1 \end{cases}$$

Using the function-space view of Gaussian Processes, we write f_π as the model for parameter π, and we can define this model f_π with the appropriate mean function and kernel function accordingly in a Gaussian Process. f_π is known as a *latent function*.

$$f_\pi \sim GP\left(0, g(K_A, K_X)\right)$$

We denote K_A as the kernel function corresponding to data where an action was sent and K_X as the kernel function corresponding to data where we only contain the main effects. The final kernel is a function g of that is a function of both kernel K_A and K_X.

When action is not applied, the kernel function g becomes a function of only the main effects X. We can write the kernel function as $g(0, K_X)$ when action is not applied, in other words, when $A = 0$.

In summary, we use the following assumptions in our GP model:

1. $Y_i \mid A_i, X_i \sim \text{Bern}\left(\pi\left(A_i, X_i\right)\right)$
2. $\pi\left(A_i, X_i\right) = \text{expit}(f_\pi\left(A_i, X_i\right))$
3. $f_\pi \sim GP\left(0, g\left(K_A, K_X\right)\right)$
4. $f_\pi | A = 0 \sim GP\left(0, K_X\right)$

Logit Transformation

Why can we not directly fit the π parameter by a Gaussian Process? This is because we recall a Gaussian Process describes Gaussian variables, which take on continuous values between $-\infty$ and $+\infty$. Probability π is a continuous value, but it is bounded between 0 and 1. Therefore, to transform a range of values $z \in (-\infty, +\infty)$ to $f(z) \in [0, 1]$ using a transformation f, an *expit* transformation is applied. The expit function is defined below:

$$\text{expit}(z) = \frac{1}{1 + e^{-z}}$$

Graphically, we can visualize the expit function approaching 0 when $z \to -\infty$, and then approaching 1 when $z \to +\infty$. When $z = 0$, the expit function is exactly $\frac{1}{2}$.

The expit function is also known as a *logistic response function*. This is because the inverse of the expit function is the logit function, which is typically used to map values from $[0, 1]$ to $(-\infty, +\infty)$.

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right)$$

In context of our Gaussian Process model on π, we mentioned one of our assumptions is that $\pi\left(A_i, X_i\right) = \text{expit}\left(f_\pi\left(A_i, X_i\right)\right)$. This necessarily implies that $\text{logit}(\pi\left(A_i, X_i\right)) = f_\pi\left(A_i, X_i\right)$.

$$\begin{aligned}
& \pi = \text{expit}(f_\pi) \\
\implies \quad & \pi = \frac{1}{1+e^{-f_\pi}} \\
\implies \quad & \frac{1}{p} = 1 + e^{-f_\pi} \\
\implies \quad & \frac{1-p}{p} = e^{-f_\pi} \\
\implies \quad & \log\left(\frac{1-p}{p}\right) = -f_\pi \\
\implies \quad & \log\left(\frac{p}{1-p}\right) = f_\pi
\end{aligned}$$

Probit Transformation

It is also worth mentioning that another common choice of transformation f from $z \in (-\infty, +\infty)$ to $f(z) \in [0, 1]$ is the probit function. The probit function is the cumulative density function of a standard Gaussian distribution.

Recall that the probability density function of any Gaussian distribution where $X \sim \mathcal{N}(\mu, \sigma)$ is

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

A standard Gaussian distribution transforms the variable x into z so that $\frac{x-\mu}{\sigma} = z$. Then, z is a standard Gaussian distribution because z has mean 0 and variance 1. We can then write the probability density function of z to be the following

$$f(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2}$$

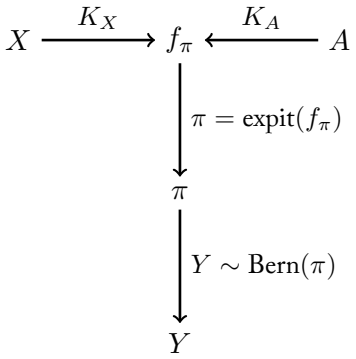
The cumulative density distribution Φ is then the sum of probabilities from the lowest possible value up to a specified value of z. Because the Gaussian distribution is continuous with zero point-mass at every point, we will integrate from negative infinite up to z.

$$\Phi(z) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2} dz$$

Naturally, $\Phi(z)$ must be in $[0, 1]$ because all probabilities must be in $[0, 1]$. A transformation using $\Phi(z)$ is known as a probit transformation.

3.12 DAG Representation

To bring a more visual interpretation to our Gaussian Process formulation, we once again turn towards a directed acyclic graph. We can represent the inputs X and A as source nodes which affects our model of the π parameter, which then affects our target Y.



Under this directed acyclic graph, we can follow the DAG to Probability algorithm on page 38 to relate the joint probability with the correct marginal and conditional probabilities.

$$\begin{aligned} f(y, \pi, f_\pi, A, X) &= f(A)f(X)f(f_\pi|X, A)f(\pi|f_\pi)f(y|\pi) \\ &= f(A)f(X)f(f_\pi|X, A)f(y|\pi) \\ &= f(A)f(X)f(f_\pi|X, A)f(y|f_\pi) \end{aligned}$$

We can point out that the relationship between π and f_π is deterministic. Once we know f_π's value, we just need to apply $\text{expit}(f_\pi) = \pi$. Thus, due to this deterministic

relationship, $f(\pi|f_\pi) = 1$. In the final step we made the direct substitution between deterministic variables expit$(f_\pi) = \pi$ to get $f\left(y|\pi\right) = f\left(y \mid f_\pi\right)$.

One term in the probability expansion looks jarring. What does it mean to have $f(f_\pi|X, A)$?

Recall that we defined $f_\pi \sim GP\left(0, g\left(K_A, K_X\right)\right)$ so f_π is a random variable as much as a function $f_\pi(A, X)$. Recall that a random variable is function that maps a sample space to real numbers. We can think of $f_\pi(A, X)$ as the function mapping a more structured sample space defined by A, X in addition to $f_\pi \sim GP(0, g(K_A, K_X))$ as a random variable being structured by the kernels of A and X. In both cases, A and X are inputs that affect the structure of f, and we are merely using two different representations to convey the same thing.

Because f_π is a random variable, then the intepretation of $f(f_\pi|X, A)$ has much of the same interpretation as any conditional probability.

3.13 Likelihood Function

Once we have our Gaussian Process formulation, the next step is to define the likelihood function. When writing out the likelihood function, we want to be explicit in defining f_π as a Gaussian Process, which is a function of a vector of random variables indexed by time. The convention we have been employing thus far has been to bold vectors and matrices, but we want to distinguish functions of vectors and matrices from values and matrices themselves. Therefore, we will place a vector above $\vec{f_\pi}$ to emphasize that $\vec{f_\pi}$ is a function of a vector of values ordered by time, so that vector $\vec{f_\pi}$ is equal to $\langle f_\pi\left(A_1, x_1\right), \cdots, f_\pi\left(A_i, x_i\right)\rangle$.

The likelihood function, as we recall, is the probability of observing the data conditioned on the parameters of the model we've specified. The parameter of $\boldsymbol{Y}$ is Bern(π) where logit$(\pi) = \vec{f_\pi} \sim GP(0, g\left(K_A, K_X\right))$. For now, we don't need to worry about the parameterization of $\vec{f}$, thinking more on a high level that $\vec{f}$ is a random variable which describes the specific probability instance π_i for observation Y_i based on specified A_i and

X_i.

$$\begin{aligned}
L(\theta) &= P\left(\boldsymbol{Y} \mid \pi\right) \\
&= P\left(\boldsymbol{Y} \mid \vec{f_\pi}\right) \\
&= \prod_{i=1}^{n} \left(1 - \text{expit}\, f_\pi\left(A_i, X_i\right)\right)^{1-Y_i} \left(\text{expit}\, f_\pi\left(A_i, X_i\right)\right)^{Y_i} \\
&= \prod_{i=1}^{n} \left(1 - \pi\left(A_i, X_i\right)\right)^{1-Y_i} \left(\pi\left(A_i, X_i\right)\right)^{Y_i} \\
&= \prod_{i=1}^{n} \left(1 - \theta\left(A_i, X_i\right)\right)^{1-Y_i} \left(\theta\left(A_i, X_i\right)\right)^{Y_i}
\end{aligned}$$

We took some more steps than neccessary in writing our likelihood function in order to provide a better idea of the generalization of Gaussian Processes on parameter θ. under weight-space view and function space view.

1. θ is a transformed linear combination of A_i, X_i. $\theta(A_i, X_i) = g(\beta_A A_i + \beta_X X_i)$ where g is some function transformation of the linear combination of A_i, X_i. This is in accordance with the *weight-space view*.
2. θ is a random variable. Specifically, $f(\theta) \sim GP(0, g(A, X))$ for some function f on parameters. This is in accordance with the *function space view*.

3.14 Prior Distribution

Next, we place a prior distribution for our parameter π to be p_π. Traditionally, the symbol π is used as the prior, but we already have π as our probability parameter characterizing the Bernoulli distribution, so we will use the notation p_π as the prior. The interpretation here is that before any data has been collected, $\pi = p_\pi$.

The prior for θ is

$$\begin{aligned}
P(\theta) &= p_\pi \\
&= \text{expit}(f_{p_\pi}) \\
&= P(\vec{f_\pi})
\end{aligned}$$

Traditionally, the prior f_{p_π} is fit on some test data. For now, we keep the prior a simple expression

3.15 Posterior Distribution

The posterior distribution is the product of likelihood and prior.

$$\begin{aligned} P(\vec{f_\pi} \mid \boldsymbol{Y}, \boldsymbol{X}, \boldsymbol{A}) &= \frac{P(\boldsymbol{Y}|\vec{f_\pi}, \boldsymbol{X}, \boldsymbol{A})P(\vec{f_\pi})}{P(\boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{A})} \\ &\propto P\left(\boldsymbol{Y} \mid \vec{f_\pi}, \boldsymbol{X}, \boldsymbol{A}\right) P\left(\vec{f_\pi}\right) \\ &= P\left(\boldsymbol{Y} \mid \vec{f_\pi}\right) P\left(\vec{f_\pi}\right) \\ &= \prod_{i=1}^{n} \left(1 - \operatorname{expit} f_\pi\left(A_i, X_i\right)\right)^{1-Y_i} \left(\operatorname{expit} f_\pi\left(A_i, X_i\right)\right)^{Y_i} f_{p_\pi} \end{aligned}$$

3.16 Marginal Likelihood

The marginal likelihood is the product of likelihood and prior integrated over all possible Gaussian processes. The marginal likelihood function here cannot be computed *analytically* because the likelihood function is not a Gaussian distribution and the principle of Normal-Normal conjugacy, as derived on page 21, does not apply. For there to exist an analytic solution, we can express the marginal likelihood as a mathematical expression that can easily be calculated.

However, we will still express a general representation of the marginal likelihood according the principles of Bayesian inference.

$$\begin{aligned} f\left(\boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{A}, p_\pi\right) &= \int_{\vec{f_\pi} \in \Omega} P\left(\vec{f_\pi} \mid \boldsymbol{Y}, \boldsymbol{X}, \boldsymbol{A}, p_\pi\right) d\vec{f_\pi} \\ &= \int_{\vec{f_\pi} \in \Omega} L(\theta) \times P(\theta) d\vec{f_\pi} \\ &= \int_{\vec{f_\pi} \in \Omega} \prod_{i=1}^{n} \left(1 - \operatorname{expit} f_\pi\left(A_i, X_i\right)\right)^{1-Y_i} \left(\operatorname{expit} f_\pi\left(A_i, X_i\right)\right)^{Y_i} f_{p_\pi} d\vec{f_\pi} \end{aligned}$$

Using the marginal likelihood, we can find the optimal $\hat{p}_\pi$ by finding the optimal $\hat{f}_{p_\pi}$ which maximizes the marginal likelihood. $\hat{p}_\pi$ is known as the maximum likelihood estimator (MLE) for the prior probability p_π. However, remember that $\hat{p}_\pi = \operatorname{expit}(f_{p_\pi})$ so the problem of optimizing p_π really become finding the best Gaussian Process fitted on prior data. Then, as new data flows in, the posterior π becomes the new prior p_π.

3.17 Predictive Distribution

Using the marginal likelihood, we can find a new posterior distribution given new data. Let $\boldsymbol{X}_*$ be new main effects data and $\boldsymbol{A}_*$ be new action data. Note that the convention we employ is to subscript $*$ as new data. If there are m new observations, we can simply append it to the column vectors $\boldsymbol{X}$ and $\boldsymbol{A}$

$$\boldsymbol{X} = \begin{bmatrix} x_1^\top \\ \vdots \\ x_n^\top \end{bmatrix}, \quad \boldsymbol{X}_* = \begin{bmatrix} x_{n+1}^\top \\ \vdots \\ x_{n+m}^\top \end{bmatrix}, \quad \boldsymbol{A} = \begin{bmatrix} A_1^\top \\ \vdots \\ A_n^\top \end{bmatrix}, \quad \boldsymbol{A}_* = \begin{bmatrix} A_{n+1}^\top \\ \vdots \\ A_{n+m}^\top \end{bmatrix}$$

Now we can interpret $P\left(f_\pi\left(\boldsymbol{X}_*, \boldsymbol{A}_*\right) \mid \boldsymbol{X}, \boldsymbol{A}, \boldsymbol{Y}, \boldsymbol{X}_*, \boldsymbol{A}_*, \hat{p}_\pi\right)$ as our generative model for π distributed according to $\boldsymbol{X}, \boldsymbol{A}, \boldsymbol{Y}, \boldsymbol{X}_*, \boldsymbol{A}_*, \hat{p}_\pi$.

The posterior distribution gives us the generative model of π, with which we could use to predict new observations of $\boldsymbol{Y}_*$.

$$\pi_* = P\left(\boldsymbol{Y}_* = 1 \mid \boldsymbol{X}, \boldsymbol{A}, \boldsymbol{Y}, \boldsymbol{X}_*, \boldsymbol{A}_*\right)$$

To do so, we use our previously defined $P\left(f_\pi\left(\boldsymbol{X}_*, \boldsymbol{A}_*\right) \mid \boldsymbol{X}, \boldsymbol{A}, \boldsymbol{Y}, \boldsymbol{X}_*, \boldsymbol{A}_*\right)$. As stated before, The vector $\vec{f_\pi}$ is equal to $\langle f_\pi\left(A_1, X_1\right), \cdots, f_\pi\left(A_n, X_n\right)\rangle$. We also have the prior values p_π.

$$\begin{aligned}
P(\boldsymbol{Y}_* \mid \boldsymbol{X}, \boldsymbol{A}, \boldsymbol{X}_*, A_*) &= \int_{-\infty}^{\infty} P\left(\boldsymbol{Y}_* \mid \vec{f_\pi}, \boldsymbol{X}_*, \boldsymbol{A}_*\right) P\left(\vec{f_\pi} \mid \boldsymbol{X}, \boldsymbol{A}\right) d\vec{f_\pi} \\
&= \int_{-\infty}^{\infty} \prod_{i=1}^{m} P\left(Y_{*i} \mid f_\pi\left(X_{*i}\right), X_{*i}, A_{*i}\right) P\left(\vec{f_\pi} \mid \boldsymbol{X}, \boldsymbol{A}\right) d\vec{f_\pi} \\
&= \int_{-\infty}^{\infty} \prod_{i=1}^{m} \left(1 - \pi\left(A_{*i}, X_{*i}\right)\right)^{1-Y_{*i}} \left(\pi\left(A_{*i}, X_{*i}\right)\right)^{Y_{*i}} f_{p_\pi} f_{p_\pi} d\vec{f_\pi} \\
&= \int_{-\infty}^{\infty} \prod_{i=1}^{m} \left(1 - \text{expit} f_\pi\left(A_{*i}, X_{*i}\right)\right)^{1-Y_{*i}} \left(\text{expit} f_\pi\left(A_{*i}, X_{*i}\right)\right)^{Y_{*i}} f_{p_\pi} d\vec{f_\pi}
\end{aligned}$$

From here, we have come full circle, relate our generative model back to the representation of our Bernoulli distribution:

$$\boldsymbol{Y}_* \,|f_\pi(\boldsymbol{A}_*, \boldsymbol{X}_*), \boldsymbol{X}, \boldsymbol{A} \sim \text{Bern}\left(\underbrace{\text{expit}\left(f_\pi(\boldsymbol{X}_*, \boldsymbol{A}_*)\right)}_{\pi(X_*, A_*)}\right)$$

Our prediction of $\pi(X_*, A_*)$ is equivalently the probability of test outcomes succeeding, primarily $Y_* = 1$, after inputting new data X_* and A_*.

After deciding the new π values for new data, X_*, A_*, Y_*, we can repeat our calculations of the likelihood function, prior distribution, and posterior distribution by using the fitted π function produced by our Gaussian process as the prior function for new data coming in over time, thus repeating the cycle of marginalization ad infinitum.

3.18 Time-Varying Effects

The motivation for such a Gaussian Process model is the obvious flexibility with which the model updates the parameter π as data is being collected over time. Other than the assumption of the target following a Bernoulli distribution, there does not exist any other assumptions about the environment or the treatment effect, which makes the Gaussian Process model highly generalizable in any binary outcome study.

Despite not making any assumptions about the environment or the treatment effect, any changes in the environment or the treatment over time will be reflected in changing values of the π as fitted under the Gaussian Process model, which makes the Gaussian Process model highly sensitive to time-varying effects. A sensitive model to time-varying effects has tremendous value in causal inference because it is able to better capture the true state of the target at any given state in time, therefore providing greater precision and richness in describing of treatment effects and main effects on the target.

However, while the Gaussian Process provides an extremely flexible and generalizable fit of the parameters of binary outcomes, the computation of Gaussian Process for binary outcomes will not be trivial because as we've noted, the marginal likelihood function cannot be solved analytically. The next section explores different options of computation in best solving the problem of marginalization.

4

Computation

If mathematical models are the language with which we describe the world, then computation is the tool with which we develop greater fluency in describing the world.

4.1 Analytic Solutions

Recall that we previously defined *analytic* solutions to be solutions that can be solved mathematically in closed-form.

We've seen a good example of an analytic solution and a non-analytic solution when solving the marginal distribution of parameters under a Gaussian regression setting and a Gaussian binary outcomes setting, respectively.

Recall that in the regression setting, the data $\boldsymbol{X}, \boldsymbol{Y}$ are both Gaussian-distributed. Because the parameters are also Gaussian distributed, the marginalization of the posterior distribution over all possible parameters yields a solution where the mean parameter can be computed as a series of matrix operations, and the covariance a series of matrix

operations as well. Namely, the mean is $\frac{1}{\sigma_0^2}\mathbf{x}_* A^{-1} X\mathbf{y}$ and the covariance is $\mathbf{x}_*^\top A^{-1}\mathbf{x}_*$.

$$\begin{aligned} f\left(\boldsymbol{Y}_* \mid \beta, \mathbf{x}_*, X, \mathbf{y}\right) &= \int P\left(Y_* \mid \mathbf{x}_*, \boldsymbol{\beta}\right) P(\boldsymbol{\beta} \mid X, \mathbf{y}) d\boldsymbol{\beta} \\ &= \int \mathbf{x}_* \boldsymbol{\beta} P(\boldsymbol{\beta} \mid X, \mathbf{y}) d\boldsymbol{\beta} \\ &= \mathcal{N}\left(\frac{1}{\sigma_0^2}\mathbf{x}_* A^{-1} X\mathbf{y}, \mathbf{x}_*^\top A^{-1}\mathbf{x}_*\right) \end{aligned}$$

In contrast, in the binary outcomes setting, the target $\boldsymbol{Y}$ is Bernoulli-distributed. By fitting a Gaussian Process on the probability parameter π of that Bernoulli distribution, the marginal likelihood can best be simplified as an integration which cannot be serialized into clear matrix operations. There is no analytic solution under the binary outcomes setting.

$$\begin{aligned} P\left(\boldsymbol{Y}_* \mid \boldsymbol{X}, \boldsymbol{A}, \boldsymbol{X}_*, A_*\right) &= \int_{-\infty}^{\infty} P\left(\boldsymbol{Y}_* \mid \vec{f_\pi}, \boldsymbol{X}_*, \boldsymbol{A}_*\right) P\left(\vec{f_\pi} \mid \boldsymbol{X}, \boldsymbol{A}\right) d\vec{f_\pi} \\ &= \int_{-\infty}^{\infty} \prod_{i=1}^{m} P\left(Y_{*i} \mid f_\pi\left(X_{*i}\right), X_{*i}, A_{*i}\right) P\left(\vec{f_\pi} \mid \boldsymbol{X}, \boldsymbol{A}\right) d\vec{f_\pi} \\ &= \int_{-\infty}^{\infty} \prod_{i=1}^{m} \left(1 - \pi\left(A_{*i}, X_{*i}\right)\right)^{1-Y_{*i}} \left(\pi\left(A_{*i}, X_{*i}\right)\right)^{Y_{*i}} f_{p_\pi} f_{p_\pi} d\vec{f_\pi} \\ &= \int_{-\infty}^{\infty} \prod_{i=1}^{m} \left(1 - \operatorname{expit} f_\pi\left(A_{*i}, X_{*i}\right)\right)^{1-Y_{*i}} \left(\operatorname{expit} f_\pi\left(A_{*i}, X_{*i}\right)\right)^{Y_{*i}} f_{p_\pi} d\vec{f_\pi} \end{aligned}$$

When there exists no analytic solutions, *approximate* solutions are the next best thing.

Computational tools are useful for finding a solution both analytically and approximately. While analytic solutions are more elegant for us to work with because of its direct mathematical representation, to the computer, both analytic and approximate solutions are just as good as long as there exists a clear, working algorithm to describe the problem-solving procedures. *Algorithms* are procedures capable of being broken down into units of computation and units of storage.

4.2 Computational Complexity

A working algorithm is good as long as it's computatable, that is, the algorithm can be broken down into a series of operations which are guaranteed to terminate.

However, sometimes it's not enough to describe the goodness of an algorithm by whether the algorithm is computable. Some algorithms may take longer than other algorithms to perform the same task, in which case, it's clearly better to use the algorithm that takes less time. To describe the degree as to which algorithms take longer, we introduce the concept of *computational complexity*.

Computational complexity describes the worst-case runtime of a series of operations. Computational complexity is typically expressed as a function f which maps a given input n to the amount of resources needed to compute n, denoted $f(n)$.

Typically, the resource of interest is the time needed to run an algorithm, so $f(n)$ is used to describe the *running time*, the number of steps needed per unit of input. Because each step is assumed to atomically take one unit of time, there exists a direct one-to-one relationship between the total number of steps in an algorithm and the total amount of time in an algorithm.

For any complex algorithm, it is simply too much book-keeping to describe the exact number of steps. Rather, it is just as useful to describe the rate of growth of the steps of an algorithm as the input n increases. This is because any algorithm is bottlenecked by the longest step which interacts with the input of length n. To describe the efficiency of any algorithm then just requires describing the rate of growth of that bottleneck as n increases.

To do so, the notation $\mathcal{O}(g(n))$ is used, where $\mathcal{O}$ denotes the asymptotic upper bound on the rate of growth of an algorithm. We say that $f(n) = \mathcal{O}(g(n))$ if there exists a constant $\varepsilon > 0$ and a number N such that for all $n > N$:

$$f(n) \leq \varepsilon g(n)$$

From here, we can conclude that any algorithm that takes a polynomial number of steps is bounded by the term with the highest order.

$$a_p n^p + a_{p-1} n^{p-1} + \cdots + a_0 = \mathcal{O}(n^p)$$

This is because there will always exist a constant $\varepsilon > 0$ and large number N such that for all $n > N$, $f(n) \leq \varepsilon g(n)$. One could show this result by setting $\varepsilon = a_p + 1$.

$$\begin{aligned}
\lim_{n \to \infty} \frac{a_p n^p + a_{p-1} n^{p-1} + \cdots + a_0}{\varepsilon n^p} &= \lim_{n \to \infty} \frac{a_p n^p}{\varepsilon n^p} \\
&= \frac{a_p}{a_p + 1} \\
&< 1
\end{aligned}$$

Therefore, for algorithms whose running time can be expressed in a polynomial, the computational complexity is $\mathcal{O}(n^p)$ where p is the highest order, or *degree*, of the polynomial. The lower the value of p, the more *efficient* the algorithm.

4.3 Matrix Operations

Having formalized computational complexity using big-O notation, we can proceed to analyze the running time of matrix operations.

Recall that under big-O notation, $\mathcal{O}(g(n))$ describes the upper bound of an algorithm with respect to input of size n. Throughout our discussion of Gaussian Processes, the inputs have always existed as data that have been organized in a matrix or column vector. Because all computation is performed with respect to a design matrix, any analysis of computational complexity should be performed with respect to this design matrix. Recall the design matrix $\boldsymbol{X}$ contains a dimension of $n \times p$ (n rows for the number of observations and p columns for the number of features).

$$\boldsymbol{X} = \begin{bmatrix} X_{11} & \dots & X_{1j} & \dots & X_{1p} \\ \vdots & & \vdots & & \vdots \\ X_{i1} & \dots & X_{ij} & \dots & X_{ip} \\ \vdots & & \vdots & & \vdots \\ X_{n1} & \dots & X_{nj} & \dots & X_{np} \end{bmatrix}$$

An algorithm that will just read in values from the design matrix will have runtime $\mathcal{O}(np)$ because the algorithm will need to have at least a single pass through all np elements inside the matrix.

An algorithm that tries to multiply the design matrix with its transpose to obtain $\boldsymbol{X}\boldsymbol{X}^\top$ using the traditional matrix multiplication approach will have runtime that is $\mathcal{O}(n^2p)$ because each element in the new matrix $\boldsymbol{X}\boldsymbol{X}^\top$ is a dot product between two vectors of length p, so performing the dot product is $\mathcal{O}(p)$. There are n^2 elements in the matrix $\boldsymbol{X}\boldsymbol{X}^\top$ and therefore the runtime is $\mathcal{O}(n^2p)$.

Similarly, an algorithm that tries to multiply the transpose of the design matrix with itself to obtain $\boldsymbol{X}^\top\boldsymbol{X}$ using the traditional matrix multiplication approach will have runtime that is $\mathcal{O}(np^2)$ because each element in the new matrix $\boldsymbol{X}^\top\boldsymbol{X}$ is a dot product between two vectors of length n, so performing the dot product is $\mathcal{O}(n)$. There are p^2 elements in the matrix $\boldsymbol{X}^\top\boldsymbol{X}$ and therefore the runtime is $\mathcal{O}(np^2)$.

4.4 Kernel Operations

Recall that the kernel, or covariance function, is a function of random variables, outputting a covariance matrix once the random variables have crystallized into observations.

In our definition of a valid kernel function on page 25, we stated that the covariance matrix produced by the kernel using the observations as inputs is *positive semi-definite*. The mathematical justification is that kernel functions must exist as inner products between random variables under any Hilbert space. In other words, the kernel functions should exist as a measure of distance between two variables, and distance cannot be negative.

A matrix Σ of dimension $n \times n$ is *positive semi-definite* if for all vectors $\mathbf{v} \in \mathbb{R}^n$, $Q(\mathbf{v}) = \mathbf{v}^\top K\mathbf{v} \geq 0$. A stronger property is *positive-definite* where for all vectors $\mathbf{v} \in \mathbb{R}^n$ and $\mathbf{v} \neq \mathbf{0}$, $Q(\mathbf{v}) = \mathbf{v}^\top K\mathbf{v} > 0$.

Alternatively, another definition typically used in a linear algebra context is that a matrix Σ is *positive semi-definite* if it is a symmetric matrix with all nonnegative eigenvalues $\lambda \geq 0$. A *positive definite* matrix will have all positive eigenvalues $\lambda > 0$.

To find all the eigenvalues, one would need to find all n solutions a characteristic equation.

$$\det(\Sigma - \lambda I)x = 0 \implies a_n\lambda^n + a_{n-1}\lambda^{n-1} + \cdots + a_0 = 0$$

Alternatively, we can check *positive-definiteness* in a matrix by computing the *Cholesky decomposition*.

Cholesky Decomposition

The *Cholesky decomposition* decomposes a positive semi-definite matrix into a lower-triangular matrix and a conjugate transpose. That is, for any positive semi-definite matrix Σ, there exists a lower-triangular matrix L such that the following relationship holds:

$$\Sigma = LL^\top$$

Lower triangular matrix L can be written as follows:

$$L = \begin{bmatrix} \ell_{1,1} & & & & 0 \\ \ell_{2,1} & \ell_{2,2} & & & \\ \ell_{3,1} & \ell_{3,2} & \ddots & & \\ \vdots & \vdots & \ddots & \ddots & \\ \ell_{n,1} & \ell_{n,2} & \dots & \ell_{n,n-1} & \ell_{n,n} \end{bmatrix}$$

The runtime for computing the Cholesky decomposition is typically $\mathcal{O}(n^3)$, where n is the dimension of the columns or the dimension of rows. Either is fine because Σ must be a square matrix in order to be positive semi-definite. A matrix is *square* if the dimensions are $n \times n$.

The runtime for the Cholesky algorithm is $\mathcal{O}(n^3)$ because it is incorporates similar techniques to Gaussian elimination. For every column i, there will be a maximum of $\frac{n(n-1)}{2} = \mathcal{O}(n^2)$ operations among the n rows via swapping, multiplication, or addition so that the i-th column has leading 1's in Gaussian elimination. Then, going through each of the n columns, there exists $\mathcal{O}(n^3)$ runtime.

Cholesky Algorithm

Algorithm 2: Cholesky Algorithm

Initialize $\mathbf{\Sigma}^{(1)} := \mathbf{\Sigma}$

Loop $\forall i \in [1, \cdots, n]$:

$$\mathbf{\Sigma}^{(i)} = \begin{pmatrix} \mathbf{I}_{i-1} & 0 & 0 \\ 0 & \Sigma_{i,i} & \mathbf{b}_i^* \\ 0 & \mathbf{b}_i & \mathbf{B}^{(i)} \end{pmatrix}$$

where I_{i-1} denotes the identity matrix of dimension $i-1$

$$\text{Initialize } \mathbf{L}_i := \begin{pmatrix} \mathbf{I}_{i-1} & 0 & 0 \\ 0 & \sqrt{\Sigma_{i,i}} & 0 \\ 0 & \frac{1}{\sqrt{\Sigma_{i,i}}}\mathbf{b}_i & \mathbf{I}_{n-i} \end{pmatrix}$$

$$\text{Update } \mathbf{\Sigma}^{(i+1)} = \begin{pmatrix} \mathbf{I}_{i-1} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \mathbf{B}^{(i)} - \frac{1}{\Sigma_{i,i}}\mathbf{b}_i\mathbf{b}_i^* \end{pmatrix}$$

where $\mathbf{\Sigma}^{(i)} = \mathbf{L}_i \mathbf{\Sigma}^{(i+1)} \mathbf{L}_i^*$

Set $\mathbf{L} := \mathbf{L}_1 \mathbf{L}_2 \dots \mathbf{L}_n$

Return $\mathbf{L}$

At each step of the Cholesky Algorithm, the operation is bottlenecked by the operation $\mathbf{B}^{(i)} - \frac{1}{\Sigma_{i,i}}\mathbf{b}_i\mathbf{b}_i^*$, which takes $\mathcal{O}(n^2)$. Since there are n iterations, the total runtime is $\mathcal{O}(n^3)$.

4.5 Approximation Algorithms

Having formalized computational complexity and familiarized ourselves with the runtime of matrix operations, we can return to solving the problem of fitting a Gaussian Process on the probability parameter π of a Bernoulli distribution, as the marginal likelihood of binary outcomes from fitted values cannot be solved analytically. For this problem, approximation algorithms are used instead.

The essence of an approximation algorithm is to repeat steps until the difference between the approximate solution satisfies the specifications of the problem within a margin of error.

Unlike deterministic algorithms, it is hard to bound the runtime of approximation algorithms because it is hard to bound the number of steps it takes for *convergence*, defined to be when the approximate solution satisfies the specifications of the problem within a margin of error.

Instead, experiments are needed to demonstrate the performance of approximation algorithms, usually with respect to a certain input. The problem that we face is identifying a good enough integration of the marginal likelihood of binary outcomes, where the input is a provided design matrix and target observations.

$$\int_{-\infty}^{\infty} \prod_{i=1}^{m} \left(1 - \operatorname{expit} f_\pi\left(A_{*i}, X_{*i}\right)\right)^{1-Y_{*i}} \left(\operatorname{expit} f_\pi\left(A_{*i}, X_{*i}\right)\right)^{Y_{*i}} f_{p_\pi} d\vec{f}$$

If we were to have found a good approximation algorithm for this integration, in the sense that it takes relatively few steps to achieve convergence, we cannot necessarily generalize the approximation algorithm is good enough to solve all other marginalization problems where the input or the length of input n changes. In contrast, a deterministic algorithm is always guaranteed to perform well with respect to the input, no matter what choice of inputs and what length of input n.

4.6 P vs. NP

Approximation algorithms are commonly used to solve NP-hard problems, which are problems that take polynomial time to verify correctness. In contrast, P problems are problems that take polynomial time to solve correctly. While there has been no proof yet that $P = NP$ nor is there proof that $P \neq NP$, the lack of deterministic polynomial algorithms and the reliance on approximation algorithms to solve NP-hard problems support the idea that $P \neq NP$.

In solving the marginalization problem using computation, we can examine three approaches: Laplace Approximation, Expectation-Propogation, and Markov chain Monte Carlo.

From here, we will employ the notation $\mathbf{f}$ in lieu of f_π so as to generalize the algorithms for any parameters.

4.7 Laplace Approximation

Performing a Laplace Approximation consists of performing a second order Taylor expansion of $\log p(\mathbf{f}|X, y)$ around the maximum of the posterior, denoted here as $q(\mathbf{f} \mid X, \mathbf{y})$:

$$q(\mathbf{f} \mid X, \mathbf{y}) = \mathcal{N}\left(\mathbf{f} \mid \hat{\mathbf{f}}, A^{-1}\right) \propto \exp\left(-\frac{1}{2}(\mathbf{f} - \hat{\mathbf{f}})^\top A(\mathbf{f} - \hat{\mathbf{f}})\right)$$

where $\hat{f} = \arg\max_f p(f|X, y)$ and A is the Hessian of the negative log posterior at the point $\hat{f}$. Explicitly, A is written as

$$A = -\nabla\nabla \log p(\mathbf{f} \mid X, \mathbf{y})$$

Derivation

Under Bayes' Theorem, the posterior is $p(\mathbf{f} \mid X, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|X)}{p(\mathbf{y}|X)}$ and the marginal distribution $p(\mathbf{y} \mid X)$ is independent of f.

We take the logarithm of the posterior. The rationale for taking logs is so that the likelihood function can be expressed in sums as opposed to products, thus enabling easier

computation. Moreover, we observe that optimal values of $\hat{\mathbf{f}}$ along $p(\mathbf{f} \mid X, \mathbf{y})$ will stay optimal under $\log p(\mathbf{f} \mid X, \mathbf{y})$ and vice versa.

We can then perform the second-order Taylor expansion as follows. The Taylor expansion gives us a good approximation of any function as a polynomial with higher order terms. However, higher order terms tend to converge to zero because the denominator is a factorial function, thereby scaling quicker than the denominator of higher-order powers.

$$\begin{aligned}\log p(\mathbf{f} \mid X, \mathbf{y}) &= \log p(\mathbf{y} \mid \mathbf{f}) + \log p(\mathbf{f} \mid X) \\ &= \log p(\mathbf{y} \mid \mathbf{f}) - \frac{1}{2}\mathbf{f}^\top K^{-1}\mathbf{f} - \frac{1}{2}\log|K| - \frac{n}{2}\log 2\pi\end{aligned}$$

To find optimal values, we are looking for values of $\mathbf{f}$ which are a criticial point, having no derivative. Taking the first derivative of $\log p(\mathbf{f} \mid X, \mathbf{y})$ with respect to f, we have

$$\nabla \log p(\mathbf{f} \mid X, \mathbf{y}) = \nabla \log p(\mathbf{y} \mid \mathbf{f}) - K^{-1}\mathbf{f}$$

We can verify the second derivative of $\log p(\mathbf{f} \mid X, \mathbf{y})$ with respect to f is negative, indicating maximality at $\mathbf{f}$.

$$\nabla\nabla \log p(\mathbf{f} \mid X, \mathbf{y}) = \nabla\nabla \log p(\mathbf{y} \mid \mathbf{f}) - K^{-1}$$

If we were to solve for maximum $\hat{\mathbf{f}}$, we would set $\hat{\mathbf{f}} = K(\nabla \log p(\mathbf{y} \mid \hat{\mathbf{f}}))$ but because $\nabla \log p(\mathbf{y} \mid \hat{\mathbf{f}})$ cannot be solved analytically, Newton's method is used to approximate $\nabla \log p(\mathbf{y} \mid \hat{\mathbf{f}})$.

$$\begin{aligned}\hat{\mathbf{f}} &= \mathbf{f} - (\nabla\nabla \log p(\mathbf{y} \mid \mathbf{f}))^{-1}\nabla \log p(\mathbf{y} \mid \mathbf{f}) \\ &= \mathbf{f} + \left(K^{-1} - \nabla\nabla \log p(\mathbf{y} \mid \mathbf{f})\right)^{-1}\left(\nabla \log p(\mathbf{y} \mid \mathbf{f}) - K^{-1}\mathbf{f}\right) \\ &= \left(K^{-1} - \nabla\nabla \log p(\mathbf{y} \mid \mathbf{f})\right)^{-1}\left(-\nabla\nabla \log p(\mathbf{y} \mid \mathbf{f})\mathbf{f} + \nabla \log p(\mathbf{y} \mid \mathbf{f})\right)\end{aligned}$$

The posterior is then modeled accordingly under this numerical approximation of $\hat{\mathbf{f}}$:

$$q(\mathbf{f} \mid X, \mathbf{y}) = \mathcal{N}\left(\hat{\mathbf{f}}, \left(K^{-1} - \nabla\nabla \log p(\mathbf{y} \mid \mathbf{f})\right)^{-1}\right)$$

Algorithm 3: Laplace Approximation

Initialize $\mathbf{\Sigma}$ (covariance matrix), $\boldsymbol{Y}$ (targets), $p(\mathbf{y} \mid \mathbf{f})$ (likelihood function)
Initialize $\mathbf{f} := \mathbf{0}$
Repeat (Newton iteration)
$W := -\nabla\nabla \log p(\mathbf{y} \mid \mathbf{f})$,
$L := \text{cholesky}\left(I + W^{\frac{1}{2}}\mathbf{\Sigma}W^{\frac{1}{2}}\right)$

$$\mathbf{b} := W\mathbf{f} + \nabla \log p(\mathbf{y} \mid \mathbf{f})$$
$$\mathbf{a} := \mathbf{b} - W^{\frac{1}{2}}L^{\top} \backslash \left(L \backslash \left(W^{\frac{1}{2}}\mathbf{\Sigma}\mathbf{b}\right)\right)$$
$$\mathbf{f} := \mathbf{\Sigma}\mathbf{a}$$

until convergence with objective $-\frac{1}{2}\mathbf{a}^{\top}\mathbf{f} + \log p(\mathbf{y} \mid \mathbf{f})$
$\log q(\mathbf{y} \mid X, \theta) := -\frac{1}{2}\mathbf{a}^{\top}\mathbf{f} + \log p(\mathbf{y} \mid \mathbf{f}) - \sum_i \log L_{ii}$ return: $\hat{\mathbf{f}} := \mathbf{f}$ (post. mode) , $\log q(\mathbf{y} \mid X, \theta)$ (approx. log marg. likelihood)

An important point to note is that we set $B = I + W^{\frac{1}{2}}\Sigma W^{\frac{1}{2}}$ because it is not always guarenteed that $W^{\frac{1}{2}}\Sigma W^{\frac{1}{2}}$ is positive-definite. Adding the identity matrix ensures that $I + W^{\frac{1}{2}}\mathbf{\Sigma}\boldsymbol{W}^{\frac{1}{2}}$ is positive-definite when applying the Cholesky decomposition.

Finally, while we cannot analyze the runtime of this algorithm using big-$\mathcal{O}$ notation, since this algorithm is an approximation algorithm, we can make a statement about its convergence property, which is when
$|\frac{1}{2}\mathbf{a}^{\top}\mathbf{f} + \log p(\mathbf{y} \mid \mathbf{f})^{(i+1)} - \frac{1}{2}\mathbf{a}^{\top}\mathbf{f} + \log p(\mathbf{y} \mid \mathbf{f})^{(i)}| < \varepsilon$ for some $\varepsilon > 0$. In other words, the algorithm for Laplace approximation terminates in $i + 1$ steps once the difference between the predicted objective function under the $i + 1$ iteration and the predicted objective function under the i iteration is less than some specified ε value. In approximation algorithms, the convergence property is essential for describing runtime or whether the algorithm terminates at all.

4.8 Expectation-Propagation

The Expectation-Propagation algorithm approximates the posterior $p(\mathbf{f} \mid X, \mathbf{y})$ by $q(\mathbf{f} \mid X, \mathbf{y})$ under the following relationship:

$$\begin{aligned} q(\mathbf{f} \mid X, \mathbf{y}) &= \frac{1}{Z_{\text{EP}}} p(\mathbf{f} \mid X) \prod_{i=1}^{n} t_i \left(f_i \mid \tilde{Z}_i, \tilde{\mu}_i, \tilde{\sigma}_i^2 \right) \\ &= \mathcal{N}(\boldsymbol{\mu}, \Sigma) \end{aligned}$$

Breaking the above equation down, we identify that $p(\mathbf{f} \mid X)$ is the prior and $\prod_{i=1}^{n} t_i \left(f_i \mid \tilde{Z}_i, \tilde{\mu}_i, \tilde{\sigma}_i^2 \right)$ is the likelihood. t_i is a local likelihood of f_i around fitted $\tilde{Z}_i, \tilde{\mu}_i, \tilde{\sigma}_i^2$ values. In other words, we can approximate $p\left(y_i \mid f_i\right)$ using $t_i \left(f_i \mid \tilde{Z}_i, \tilde{\mu}_i, \tilde{\sigma}_i^2 \right)$ to obtain the following relationships:

$$\begin{aligned} p\left(y_i \mid f_i\right) &= t_i \left(f_i \mid \tilde{Z}_i, \tilde{\mu}_i, \tilde{\sigma}_i^2 \right) \\ &= \tilde{Z}_i \mathcal{N} \left(f_i \mid \tilde{\mu}_i, \tilde{\sigma}_i^2 \right) \end{aligned}$$

Finally, $Z_{\text{EP}} = q(\mathbf{y} \mid X)$, which is the approximation of $p(y|X)$.

$$\begin{aligned} Z &= p(\mathbf{y} \mid X) \\ &= \int p(\mathbf{f} \mid X) \prod_{i=1}^{n} p\left(y_i \mid f_i\right) d\mathbf{f} \end{aligned}$$

The Expectation-Propagation approach can be thought of as constantly alternating between approximating the posterior and approximating the marginal likelihood. To make the approximation of the posterior $q(\mathbf{f} \mid X, \mathbf{y})$, the Expectation-Propagation algorithm holds variables approximated from the marginal likelihood to be fixed. Likewise, when approximating the marginal likelihood $q(\mathbf{y} \mid X)$, the Expectation-Propagation algorithm holds variables approximated from the posterior to be fixed. By "propagating" these fixed values, the integration of the marginal and posterior distributions become analytic as opposed to non-analytic, thereby allowing us to identify numerical values.

Expectation-Propagation is an example of *variational inference*, which minimizes the *Kullback–Leibler divergence* between two probability distributions. The Kullback-Leibler divergence is defined to be the following integral:

$$D_{\text{KL}}(P \| Q) = \int_{-\infty}^{\infty} p(x) \log \left(\frac{p(x)}{q(x)} \right) dx$$

Derivation

Using the notation $\tilde{\mu}_i$ and $\tilde{\sigma}_i^2$ as the local likelihood approximations under the marginal distribution $\tilde{Z} = p\left(\mathbf{y}_i \mid X_i\right)$, and μ and Σ for the parameters of the approximate posterior, we can optimize the likelihoods sequentially, leaving out the i-th observation. This is often referred to as a *cavity distribution*.

$$q_{-i}\left(f_i\right) \propto \int p(\mathbf{f} \mid X) \prod_{j \neq i} L_j\left(f_j \mid \tilde{Z}_j, \tilde{\mu}_j, \tilde{\sigma}_j^2\right) df_j$$

Then, to find the exact likelihood, we multiply the cavity distribution by the likelihood of the i-th observation:

$$\hat{q}\left(f_i\right) \simeq q_{-i}\left(f_i\right) p\left(y_i \mid f_i\right)$$

The cavity distribution has the following parameters:

$$\begin{aligned} q_{-i}\left(f_i\right) &= \mathcal{N}\left(f_i \mid \mu_{-i}, \sigma_{-i}^2\right) \\ \text{where } \mu_{-i} = \sigma_{-i}^2\left(\sigma_i^{-2}\mu_i - \tilde{\sigma}_i^{-2}\tilde{\mu}_i\right), &\text{ and } \sigma_{-i}^2 = \left(\sigma_i^{-2} - \tilde{\sigma}_i^{-2}\right)^{-1}. \end{aligned}$$

Multiplying the cavity distribution and the likelihood, we have updated parameters for the marginal distribution, from which we can repeat the steps once more until convergence.

$$\begin{aligned} \tilde{\mu}_i &= \tilde{\sigma}_i^2\left(\hat{\sigma}_i^{-2}\hat{\mu}_i - \sigma_{-i}^{-2}\mu_{-i}\right), \quad \tilde{\sigma}_i^2 = \left(\hat{\sigma}_i^{-2} - \sigma_{-i}^{-2}\right)^{-1} \\ \tilde{Z}_i &= \hat{Z}_i\sqrt{2\pi}\sqrt{\sigma_{-i}^2 + \tilde{\sigma}_i^2}\exp\left(\tfrac{1}{2}\left(\mu_{-i} - \tilde{\mu}_i\right)^2 / \left(\sigma_{-i}^2 + \tilde{\sigma}_i^2\right)\right) \end{aligned}$$

Algorithm 4: Expectation-Propagation

inputs: $\mathbf{\Sigma}$ (covariance matrix), $\mathbf{y}$ (targets)
Initialize $\tilde{\boldsymbol{\nu}} := \mathbf{0}, \tilde{\boldsymbol{\tau}} := \mathbf{0}, \Sigma := K, \boldsymbol{\mu} := \mathbf{0}$
Repeat
for $i := [1, \cdots, n]$ do

$$\tau_{-i} := \sigma_i^{-2} - \tilde{\tau}_i$$
$$\nu_{-i} := \sigma_i^{-2}\mu_i - \tilde{\nu}_i$$
$$\Delta\tilde{\tau} := \hat{\sigma}_i^{-2} - \tau_{-i} - \tilde{\tau}_i \text{ and } \tilde{\tau}_i := \tilde{\tau}_i + \Delta\tilde{\tau}$$
$$\tilde{\nu}_i := \hat{\sigma}_i^{-2}\hat{\mu}_i - \nu_{-i}$$
$$\Sigma := \Sigma - \left((\Delta\tilde{\tau})^{-1} + \Sigma_{ii}\right)^{-1} \mathbf{s}_i\mathbf{s}_i^\top$$
$$\mu := \Sigma\tilde{\nu}$$

end for

$$L := \text{ cholesky } \left(I_n + \tilde{S}^{\frac{1}{2}} K \tilde{S}^{\frac{1}{2}}\right)$$
$$V := L^\top \backslash \tilde{S}^{\frac{1}{2}} K$$
$$\Sigma := K - V^\top V \text{ and } \mu := \Sigma\tilde{\nu}$$

until convergence
compute $\log Z_{\text{EP}}$ (approximate log marginal likelihood)
Return: $\tilde{\boldsymbol{\nu}}, \tilde{\boldsymbol{\tau}}$ (natural site parameters), $\log Z_{\text{EP}}$

The Expectation-Propagation algorithm has the convergence property of terminating when $|\Sigma^{(i+1)} - \Sigma^{(i)}| < \varepsilon$ and $\left|\mu^{(i+1)} - \mu^{(i)}\right| < \varepsilon$ for some $\varepsilon > 0$. In other words, the algorithm for Expectation-Propagation terminates in $i + 1$ steps once the difference between the predicted parameters of the marginal distribution under the $i + 1$ iteration and the predicted parameters of the marginal distribution the i iteration are both less than some specified ε value.

4.9 Markov chain Monte Carlo

The Markov chain Monte Carlo (MCMC) can be thought of as a brute-force solution that does not necessarily guarantee quick convergence in context of our problem.

Definition

Given a random experiment and event A, a Monte Carlo estimate of $P(A)$ is obtained by repeating a random experiment many times, taking the proportion of trials in which A occurs as an approximation for $P(A)$.

More formally, given a probability distribution π and a function f, we are want to calculate or estimate the expected value of $f(x)$ for $X \sim \pi$. We are trying to find the expectation $E[f(x)]$ but we can't evaluate $X \sim \pi$ analytically.

Instead we can construct an ergodic Markov Chain with π as the limiting distribution, also known as an *invariant*.

Markov Chain

A Markov Chain is a sequence of Random Variables that follow the Markov property:

$$P(X_N = k | X_0 = x_0, \cdots, X_{N-1} = x_{n-1}) = P(X_N = k | X_{N-1} = x_{n-1})$$

Ergodic

A Markov Chain is ergodic if it is

1. Irreducible (every state can get back to itself)
2. Aperiodic (the greatest common factor of all the steps needed to get back to the original state is 1)
3. Positive Recurrent (the expected return time to get back to the original state is not zero)

Algorithm

1. Start $X_0 = i$ for i
2. Run chain for a long time ($N >> 1$)

3. Then, we can obtain $X_i \sim \pi$
4. Treat X_{ij} to be drawn from π

In theory, we should get something that's very close to the true value π as as a result from our simulation, but it is often not clear how large N is needed to do so.

4.10 Testbed

To analyze approximation algorithms like Laplace Approximation, Expectation-Propogation, and Markov chain Monte Carlo, it is useful to create a *testbed* of data generated from a known model. The known model which generates the testbed is known as the *ground truth*.

Then to test performance of these models, one can define a loss function between the predicted values of the parameter of the model ($\hat{\theta}$) and the ground truth values of the parameter θ. The loss function is typically defined to be the L_2 loss function. More performant models have smaller L_2 values.

$$L_2 = \sum_{i=1}^{n} \left(\theta_i - \hat{\theta}_i\right)^2$$

For Gaussian Processes on binary outcomes, to model the ground truth, we can fit a Beta distribution on the probability parameter π. Then, as we model π changing over time, we can update the parameters of the Beta distribution to produce a different ground truth π parameter. The ground truth π parameter can then be compared against fitted $\hat{\pi}$ from a Gaussian Process.

Beta-Binomial Conjugacy

The Beta distribution is a good prior distribution to describe a probability parameter p. This is because we can recall the beta distribution to have the probability density function (PDF):

$$f(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\mathrm{B}(\alpha,\beta)} \text{ where } \mathrm{B}(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$$

If $\alpha = 1$ and $\beta = 1$, it is evident that the PDF of the Beta distribution becomes the PDF of the Uniform distribution from $[0, 1]$.

$$f(x) = 1$$

Thus, it is often a good choice to place a Beta prior on uniform random variables with the prior

$$p \sim \text{Beta}(a, b)$$

Then, the distribution of our binary outcomes under the probability p parameter collectively can be represented as a Binomial distribution:

$$Y \mid p \sim \text{Bin}(n, p)$$

By focusing only on the terms that involve our probability parameter p, we multiply the prior and the likelihood to get the posterior.

$$p^y(1-p)^{n-y} \cdot p^{a-1}(1-p)^{b-1} = p^{a+y-1}(1-p)^{b+n-y-1}.$$

Hence, using Bayes' rule once again, we have show that the posterior distribution is still Beta, and the parameters of the Beta update in a simple way:

$$p \mid (Y = y) \sim \text{Beta}(a + y, b + n - y)$$

4.11 Software

The Python package `GPflow` utilizes Markov chain Monte Carlo – there is no Expectation-Propagation as a Variational Inference option, nor is there Laplace approximation. In contrast, the Python package `GPy` uses Expectation-Propagation and the Python package `sklearn` uses Laplace Approximation. Some preliminary testbeds have been generated and tested against Laplace Approximation, Expectation-Propogation, and Markov chain Monte Carlo using these packages, but no conclusive commentary can be made yet about performance of Gaussian Process models on binary outcomes under these different computational approaches.

5

Flexibility

Gaussian Processes fitted on changing values of π over time provide greater *flexibility* in describing the effects of input variables on a target variable across time. This flexibility has certain desirable qualities in context of causal inference, decision theory, and bandit learning, giving us greater control and structure in how we model change across all these fields.

5.1 Decision Theory

Decision theory examines the consequences of actions along some defined loss function. Formally defined, there exists five components to decision theory:

1. Parameter θ defining the environment: The true parameter which characterizes the distribution of random variable X
2. Data D: Crystallized observations of random variable X
3. Objective $g(\theta)$: The estimand that we try to estimate. If the objective is not the true parameter itself, it is a function of the true parameter θ we are trying to estimate
4. Decision rule $\delta(X)$: How we decide to estimate $g(\theta)$

5. Loss function $L(\theta, \delta)$: The regret of our decision between our estimated $g(\theta)$ and the true value of $g(\theta)$ as chosen by our decision rule

The goal in decision theory is to identify the most optimal choice an agent can make by examining the consequences of actions, known as a decision rule, that the agent can take.

Optimal decision-making is a highly coveted skill in life, which explains for its popularity across many disciplines, from economics to psychology to medicine. By being able to break down optimal decision-making using the language of mathematics, there can be better structure in thinking about this highly complex and open-ended task.

Inference

On a fundamental level, decision theory can be perceived as an inference problem. The goal of inference is to construct good models that describe the truth based on observed data. Similarly, the goal of decision theory is to make good decisions grounded in truth based on observed data.

To make this connection more explicit, consider a decision theory problem where an agent performs an action that estimates the mean θ of the random variable X where $X \sim \mathcal{N}(\theta, 1)$ and observes a new data point x_i after performing the action. We can then define the components of our decision theory problem as follows:

1. Parameter $\theta = \mu$
2. Data $D = \{x_1, \cdots, x_n\}$ (where n is the number of observed data before performing action)
3. Objective $g(\theta) = \theta$
4. Decision rule $\delta(X) = \sum_{i=1}^{n} X$
5. Loss function $L(\theta, \delta) = (\delta(X) - \theta)^2$

Seen here, the decision rule is taking the expected value of observed data, which under long periods of time, should converge to the true mean parameter due to the law of large numbers. Thus, over long periods of time, as more data gets collected, the loss function approaches 0, illustrating that this decision achives optimality under the loss function.

5.2 Risk Function

To be more structured in how we think about a decision's goodness, we introduce a risk function $R(\theta, \delta)$, which evaluates a decision rule's success over a large number of experiments with fixed parameter θ.

If we observe X many times independently, we should expect to see the average loss function over all possible parameters θ under decision rule δ to converge to the risk $R(\theta, \delta)$:

$$\begin{aligned} R(\theta, \delta) &= \mathbb{E}[L(\theta, \delta(X))] \\ &= \int L(\theta, \delta(x)) P_\theta(dx) \end{aligned}$$

If our decision rule is not fixed, but probabilistic, we would need to average the loss function under all possible types of decisions, in addition to all possible true parameters θ as befire:

$$R(\theta, \delta) = \iint L(\theta, \delta) P_{\delta|X}(d\delta \mid x) P_\theta(dx).$$

Admissibility

The risk function gives us a useful metric in comparing different decision rules. The language for comparing two decision rules $\delta, \delta' \in \mathcal{D}$ is through admissibility.

Before defining admissibility, let's consider what it means for a decision rule δ' to be better than δ. We say that δ' *dominates* (is better) than δ if for all possible parameter values the risk function of δ' is lower than or equal to δ where at some parameter θ there exists a stricly lower risk function value at δ' compared to the risk function value at δ.

Dominating Decision Rule

δ' dominates δ if

$$\begin{aligned} \forall \theta \in \Theta : R\left(\theta, \delta'\right) &\leq R\left(\theta, \delta\right) \\ \exists \theta_0 \in \Theta : R\left(\theta_0, \delta'\right) &< R\left(\theta_0, \delta\right) \end{aligned}$$

A decision rule δ is *admissible* in the class of decision rules $\mathcal{D}$ if it is not dominated by any other decision rule in $\mathcal{D}$. In other words, there does not exist a decision rule $\delta' \in \mathcal{D}$ that dominates δ.

The opposite of admissible is *inadmissible*. A decision rule is *inadmissible* if there exists another decision rule that dominates it.

An important point of note is that admissibility is a relationship between two different decision rules from the same class $\mathcal{D}$. If there only exists one decision rule $\delta \in \mathcal{D}$, then by default δ is admissible, because there exists no other decision rules in the same class $\mathcal{D}$ that dominate δ.

Moreover, admissibility is always taken with respect to a risk function, which is the expected loss function under all possible θ. Thus, if the loss function changes, the risk function changes, and so too the properties of admissibility.

Optimality

In mathematics and statistics, we always define optimality with respect to a loss function. For example, a fitted value from our model, or an *estimate*, is optimal if it achieves the minimum value according to a defined loss function. The loss function is usually convex[1], so as to actually have a minimum value. If it were concave[2], then the function would have many minimum values that approach negative infinite, and there would be nothing to optimize for! Thus, the choice of a well-defined loss function is a major component in defining optimality.

[1] convex functions are functions whose second derivatives are positive

It may be tempting to use the risk function and definitions of admissibility to define an optimal decision rule, but it is critical to observe that admissibility does not imply optimality. All we know from our definition of admissibility is that there exists certain decision rules δ that are inadmissible (not optimal) and certain decision rules δ' that are admissible (*potentially* optimal). At best, we can winnow out the inadmissible decision rules to have ourselves a smaller class of potentially optimal, admissible decision rules.

[2] concave functions are functions whose second derivatives are negative

Optimality of a decision rule is conditional on both the loss function $L(\theta, \delta)$ and the parameter θ. It's clear that decision rules with lower loss function values are more optimal, but it's never guaranteed that there exists a globally optimal decision rule with lowest loss function value under all possible θ.

Moreover, we have to remember that optimality of a decision rule is always taken in context of a class of decision rules $\mathcal{D}$. If δ was the only decision rule in $\mathcal{D}$, then there would be no other decision rule we can follow other than δ, and therefore by default, δ must be optimal! Thus, in decision theory, we must be explicit in our definition of the state space of all decision rules $\mathcal{D}$.

Minimaxity

Some alternative metrics have emerged in decision theory to describe desirable characteristics of decision rules. One such metric is minimaxity.

The decision rule δ_M is *minimax* within a class of rules $\mathcal{D}$ if

$$\delta_M \in \arg\min_{\delta \in \mathcal{D}} \max_{\theta \in \Theta} R(\theta, \delta)$$

The interpretation of a minimax δ_M is that it is the decision rule whose largest possible risk under any parameter $\theta \in \Theta$ is the smallest out of all other decision rules in $\mathcal{D}$.

The steps for finding the decision rule δ_M that is minimax is a two-step process, as its definition implies:

1. For all decision rules $\tilde{\delta} \in \mathcal{D}$, find $\theta^*(\tilde{\delta}) := \arg\max_{\theta \in \Theta} R(\theta, \tilde{\delta})$.
2. For all pairs of of $(\tilde{\delta}, \theta^*(\tilde{\delta}))$, find $\delta_M \ni \arg\min_{\delta \in \mathcal{D}} R\left(\theta^*(\tilde{\delta}), \tilde{\delta}\right)$.

5.3 Flexibility in Decision Theory

Decision theory with fixed parameters can be limiting in many ways, because parameters in the real-world are often not fixed. Flexible modeling of the parameters changing over time is ultimately a relaxation of the assumption in decision theory that parameters must be fixed. This has the desirable quality in making any decision theory problem more generalizable to the ground truth where parameters are changing as a consequence of a changing world.

5.4 Reinforcement Learning

Reinforcement Learning can be thought of as an extension of problems derived from decision theory. While the goal of decision theory is to make optimal decisions over time, the goal of reinforcement learning is to reinforce actions that maximize rewards over time, and thus both reinforcement learning and decision theory share the same ideas in principle.

What is different between the two is that reinforcement learning adopts a different formulation of the problem, using *policies* rather than decision rules and *rewards* instead of an objective. In reinforcement learning, there is more flexibility in defining the objective (because reward functions can change over time), less assumptions about the decision rules (agents can choose to engage in on-policy learning or off-policy learning, i.e. change between decision rules), and less assumptions about the environment (the environment need not stay fixed with a true parameter).

The lack of assumptions about the decision rules and the environment speaks to the focused idea in reinforcement learning that the primary interest is the interaction between decision rules and the environment in maximizing rewards, and therefore having a model of the decision rule and the environment is a sufficient, but not necessary, condition. It is perfectly fine, and often more computationally efficient, for our agent to maximize rewards without knowing anything about the environment at all.

5.5 Bandit Learning

Bandit learning is a subproblem in the field of reinforcement learning describing an agent interacting with the environment by choosing one of k actions, after which a reward conditioned on the agent's actions at any time point (A_t) is received by the agent.

Formally, each of the k actions has an expected reward (R_t) given that that action (A_t) is selected, known as the *value*. The value then of an arbitrary action a, denoted $q_*(a)$, is the expected reward given that a is selected:

$$q_*(a) \doteq \mathbb{E}\left[R_t \mid A_t = a\right]$$

The bandit's goal in interacting with the environment is to maximize rewards. Although the bandit does not know the action values with certainty, it can estimate the value of action a at time step t by defining its own function $Q_t(a)$. A good $Q_t(a)$ function is as close to $q_*(a)$ as possible.

It is the bandit's best interests to learn $q_*(a)$, the true value function over successive interactions with the environment. With a flexible Gaussian Process model on $Q_t(a)$, the bandit can come better learn $q_*(a)$, even if the value function $q_*(a)$ changes over time.

5.6 Towards Greater Flexibility

From here, we begin to see the potential for Gaussian Processes to be rather limitless. Gaussian Processes have great flexibility in modeling parameters and functions, provided that the computation converges quickly. For future exploration, different kernels, hyperparameter-tuning, and cross-validation can all be studied, providing many more ways of increasing the flexibility of Gaussian Processes in describing changing parameters in a changing world. As computation continues to accelerate and data continues to aggregate, the flexible power of Gaussian Processes will become more pronounced, ultimately helping us to better navigate the constant change in our lives.

References

[1] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*, English. 2019, OCLC: 1178958074, ISBN: 9780262256834. [Online]. Available: http://www.vlebooks.com/vleweb/product/openreader?id=none&isbn=9780262256834 (visited on 03/26/2021).

[2] T. H. Cormen, Ed., *Introduction to algorithms*, 3rd ed. Cambridge, Mass: MIT Press, 2009, OCLC: ocn311310321, ISBN: 9780262033848 9780262533058.

[3] J. Erickson, *Algorithms*, English. 2019, OCLC: 1135409183, ISBN: 9781792644832.

[4] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction*, Second edition, ser. Adaptive computation and machine learning series. Cambridge, Massachusetts: The MIT Press, 2018, ISBN: 9780262039246.

[5] S. M. Kay, *Fundamentals of statistical signal processing*, ser. Prentice Hall signal processing series. Englewood Cliffs, N.J: Prentice-Hall PTR, 1993, ISBN: 9780133457117 9780135041352 9780132808033.

[6] R. P. Dobrow, *Introduction to stochastic processes with R*. Hoboken, New Jersey: John Wiley & Sons, 2016, ISBN: 9781118740729 9781118740705.

[7] E. P. Liski, "An introduction to categorical data analysis, 2nd edition by alan agresti," en, *International Statistical Review*, vol. 75, no. 3, pp. 414–414, Dec. 2007, ISSN: 03067734. DOI: 10.1111/j.1751-5823.2007.00030_6.x. [Online]. Available: http://doi.wiley.com/10.1111/j.1751-5823.2007.00030_6.x (visited on 03/26/2021).

[8] A. Agresti, *Foundations of linear and generalized linear models*, ser. Wiley series in probability and statistics. Hoboken, New Jersey: John Wiley & Sons Inc, 2015, ISBN: 9781118730058 9781118730300.

[9] A. N. Shiryaev, *Probability*, ser. Graduate Texts in Mathematics. New York, NY: Springer New York, 1996, vol. 95, ISBN: 9781475725414 9781475725391. DOI: 10.1007/978-1-4757-2539-1. [Online]. Available: http://link.springer.com/10.1007/978-1-4757-2539-1 (visited on 03/26/2021).

[10] I. Ntzoufras and C. Tarantola, "Conjugate and conditional conjugate Bayesian analysis of discrete graphical models of marginal independence," en, *Computational Statistics & Data Analysis*, vol. 66, pp. 161–177, Oct. 2013, ISSN: 01679473. DOI: 10.1016/j.csda.2013.04.005. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167947313001357 (visited on 03/26/2021).

[11] S. L. Zeger and K.-Y. Liang, "An overview of methods for the analysis of longitudinal data," en, *Statistics in Medicine*, vol. 11, no. 14-15, pp. 1825–1839, 1992, ISSN: 02776715, 10970258. DOI: 10.1002/sim.4780111406. [Online]. Available: http://doi.wiley.com/10.1002/sim.4780111406 (visited on 03/26/2021).

[12] L. Wasserman, *All of nonparametric statistics*, ser. Springer texts in statistics. New York: Springer, 2006, ISBN: 9780387251455.

[13] Hyun-Chul Kim and Z. Ghahramani, "Bayesian gaussian process classification with the em-ep algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 12, pp. 1948–1959, Dec. 2006, ISSN: 0162-8828, 2160-9292. DOI: 10.1109/TPAMI.2006.238. [Online]. Available: http://ieeexplore.ieee.org/document/1717455/ (visited on 03/26/2021).

[14] Fei Cheng, Jiangsheng Yu, and Huilin Xiong, "Facial expression recognition in jaffe dataset based on gaussian process classification," *IEEE Transactions on Neural Networks*, vol. 21, no. 10, pp. 1685–1690, Oct. 2010, ISSN: 1045-9227, 1941-0093. DOI: 10.1109/TNN.2010.2064176. [Online]. Available:

http://ieeexplore.ieee.org/document/5551215/ (visited on 03/26/2021).

[15] D. J. Russo, B. Van Roy, A. Kazerouni, I. Osband, and Z. Wen, *A tutorial on thompson sampling*, English. Norwell, MA: Now Publishers, 2018, OCLC: 1195825895, ISBN: 9781680834710. [Online]. Available: http://public.eblib.com/choice/PublicFullRecord.aspx?p=6309027 (visited on 03/26/2021).

[16] B. Szabó, A. W. van der Vaart, and J. H. van Zanten, "Frequentist coverage of adaptive nonparametric Bayesian credible sets," *The Annals of Statistics*, vol. 43, no. 4, Aug. 2015, ISSN: 0090-5364. DOI: 10.1214/14-AOS1270. [Online]. Available: https://projecteuclid.org/journals/annals-of-statistics/volume-43/issue-4/Frequentist-coverage-of-adaptive-nonparametric-Bayesian-credible-sets/10.1214/14-AOS1270.full (visited on 03/26/2021).

[17] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *2015 IEEE Information Theory Workshop (ITW)*, Jerusalem, Israel: IEEE, Apr. 2015, pp. 1–5, ISBN: 9781479955244 9781479955268. DOI: 10.1109/ITW.2015.7133169. [Online]. Available: http://ieeexplore.ieee.org/document/7133169/ (visited on 03/26/2021).

[18] P. Diaconis and D. Ylvisaker, "Conjugate priors for exponential families," *The Annals of Statistics*, vol. 7, no. 2, Mar. 1979, ISSN: 0090-5364. DOI: 10.1214/aos/1176344611. [Online]. Available: https://projecteuclid.org/journals/annals-of-statistics/volume-7/issue-2/Conjugate-Priors-for-Exponential-Families/10.1214/aos/1176344611.full (visited on 03/26/2021).

[19] S. Karlin, "Admissibility for estimation with quadratic loss," en, *The Annals of Mathematical Statistics*, vol. 29, no. 2, pp. 406–436, Jun. 1958, ISSN: 0003-4851. DOI: 10.1214/aoms/1177706620. [Online]. Available: http://projecteuclid.org/euclid.aoms/1177706620 (visited on 03/26/2021).

[20] H. Cramér, *Mathematical methods of statistics*, ser. Princeton landmarks in mathematics and physics. Princeton: Princeton University Press, 1999, ISBN: 9780691005478.

[21] H. van Trees, "Bounds on the accuracy attainable in the estimation of continuous random processes," en, *IEEE Transactions on Information Theory*, vol. 12, no. 3, pp. 298–305, Jul. 1966, ISSN: 0018-9448. DOI: 10.1109/TIT.1966.1053910. [Online]. Available: http://ieeexplore.ieee.org/document/1053910/ (visited on 03/26/2021).

[22] K. V. Mardia, J. T. Kent, and J. M. Bibby, *Multivariate analysis*, ser. Probability and mathematical statistics. London ; New York: Academic Press, 1979, ISBN: 9780124712508 9780124712522.

[23] I. H. Bernstein, C. P. Garbin, and G. K. Teng, *Applied multivariate analysis*, en. New York, NY: Springer New York, 1988, ISBN: 9781461387428 9781461387404. DOI: 10.1007/978-1-4613-8740-4. [Online]. Available: http://link.springer.com/10.1007/978-1-4613-8740-4 (visited on 03/26/2021).

[24] A. Belloni, V. Chernozhukov, and C. Hansen, "High-dimensional methods and inference on structural and treatment effects," en, *Journal of Economic Perspectives*, vol. 28, no. 2, pp. 29–50, May 2014, ISSN: 0895-3309. DOI: 10.1257/jep.28.2.29. [Online]. Available: https://pubs.aeaweb.org/doi/10.1257/jep.28.2.29 (visited on 03/26/2021).

[25] L. Wasserman, *All of statistics: a concise course in statistical inference*, eng, Corrected second printing, 2005, ser. Springer texts in statistics. New York, NY: Springer, 2010, OCLC: 837651382, ISBN: 9780387217369 9781441923226.

[26] S. Axler, *Linear algebra done right*. New York: Springer, 2014, ISBN: 9783319110790.

[27] C. E. Frangakis, T. Qian, Z. Wu, and I. Díaz, "Rejoinder to Discussions on: Deductive derivation and turing-computerization of semiparametric efficient estimation," en, *Biometrics*, vol. 71, no. 4, pp. 881–883, Dec. 2015, ISSN: 0006341X. DOI: 10.1111/biom.12365. [Online]. Available: http://doi.wiley.com/10.1111/biom.12365 (visited on 03/26/2021).

[28] J. M. Rehg, S. A. Murphy, and S. Kumar, Eds., *Mobile Health: Sensors, Analytic Methods, and Applications*, en. Cham: Springer International Publishing, 2017, ISBN: 9783319513935 9783319513942. DOI: 10.1007/978-3-319-51394-2. [Online]. Available: http://link.springer.com/10.1007/978-3-319-51394-2 (visited on 03/26/2021).

[29] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. New York, NY: Springer New York, 2009, ISBN: 9780387848570 9780387848587. DOI: 10.1007/978-0-387-84858-7. [Online]. Available: http://link.springer.com/10.1007/978-0-387-84858-7 (visited on 03/26/2021).

[30] J. Friedman, T. Hastie, and R. Tibshirani, "Sparse inverse covariance estimation with the graphical lasso," en, *Biostatistics*, vol. 9, no. 3, pp. 432–441, Jul. 2008, ISSN: 1465-4644, 1468-4357. DOI: 10.1093/biostatistics/kxm045. [Online]. Available: https://academic.oup.com/biostatistics/article-lookup/doi/10.1093/biostatistics/kxm045 (visited on 03/26/2021).

[31] D. M. Witten, R. Tibshirani, and T. Hastie, "A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis," en, *Biostatistics*, vol. 10, no. 3, pp. 515–534, Jul. 2009, ISSN: 1465-4644, 1468-4357. DOI: 10.1093/biostatistics/kxp008. [Online]. Available: https://academic.oup.com/biostatistics/article-lookup/doi/10.1093/biostatistics/kxp008 (visited on 03/26/2021).

[32] N. Meinshausen and P. Bühlmann, "High-dimensional graphs and variable selection with the Lasso," *The Annals of Statistics*, vol. 34, no. 3, Jun. 2006, ISSN: 0090-5364. DOI: 10.1214/009053606000000281. [Online]. Available: https://projecteuclid.org/journals/annals-of-statistics/volume-34/issue-3/High-dimensional-graphs-and-variable-selection-with-the-Lasso/10.1214/009053606000000281.full (visited on 03/26/2021).

[33] V. Ročková and E. I. George, "EMVS: The EM Approach to Bayesian Variable Selection," en, *Journal of the American Statistical Association*,

vol. 109, no. 506, pp. 828–846, Apr. 2014, ISSN: 0162-1459, 1537-274X. DOI: 10.1080/01621459.2013.869223. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/01621459.2013.869223 (visited on 03/26/2021).

[34] A. Belloni, V. Chernozhukov, and C. Hansen, "Inference on Treatment Effects after Selection among High-Dimensional Controls," en, *The Review of Economic Studies*, vol. 81, no. 2, pp. 608–650, Apr. 2014, ISSN: 0034-6527, 1467-937X. DOI: 10.1093/restud/rdt044. [Online]. Available: https://academic.oup.com/restud/article-lookup/doi/10.1093/restud/rdt044 (visited on 03/26/2021).

[35] T. W. Anderson, *An introduction to multivariate statistical analysis*, 3rd ed, ser. Wiley series in probability and statistics. Hoboken, N.J: Wiley-Interscience, 2003, ISBN: 9780471360919.

[36] E. J. Hannan, Ed., *Multiple Time Series*, en, ser. Wiley Series in Probability and Statistics. Hoboken, NJ, USA: John Wiley & Sons, Inc., Aug. 1970, ISBN: 9780470316429 9780471348054. DOI: 10.1002/9780470316429. [Online]. Available: http://doi.wiley.com/10.1002/9780470316429 (visited on 03/26/2021).

[37] G. H. Golub and C. F. Van Loan, *Matrix computations*, Fourth edition, ser. Johns Hopkins studies in the mathematical sciences. Baltimore: The Johns Hopkins University Press, 2013, OCLC: ocn824733531, ISBN: 9781421407944.

[38] N. Choudhuri, S. Ghosal, and A. Roy, "Nonparametric binary regression using a Gaussian process prior," en, *Statistical Methodology*, vol. 4, no. 2, pp. 227–243, Apr. 2007, ISSN: 15723127. DOI: 10.1016/j.stamet.2006.07.003. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1572312706000475 (visited on 03/26/2021).

[39] J. O. Berger, *Statistical Decision Theory and Bayesian Analysis*, ser. Springer Series in Statistics. New York, NY: Springer New York, 1985, ISBN: 9781441930743 9781475742862. DOI: 10.1007/978-1-4757-4286-2. [Online]. Available: http://link.springer.com/10.1007/978-1-4757-4286-2 (visited on 03/26/2021).

[40] A. G. d. G. Matthews, M. van der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman, "GPflow: A Gaussian process library using TensorFlow," *Journal of Machine Learning Research*, vol. 18, no. 40, pp. 1–6, Apr. 2017. [Online]. Available: http://jmlr.org/papers/v18/16-537.html.

[41] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[42] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[43] GPy, *GPy: A gaussian process framework in python*, http://github.com/SheffieldML/GPy, since 2012.