Modelling Bitcell Behaviour

Maria Treesa Sebastian

Supervisor : Hao Chi Kiang
Examiner : Anders Nordgaard

External supervisors : Babak Mohammadi, Tom Johansson
Upphovsrätt

Detta dokument hålls tillgängligt på Internet - eller dess framtida ersättare - under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/.

Copyright

The publishers will keep this document online on the Internet - or its possible replacement - for a period of 25 years starting from the date of publication barring exceptional circumstances.

The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/hers own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: http://www.ep.liu.se/.

© Maria Treesa Sebastian
Abstract

With advancements in technology, the dimensions of transistors are scaling down. It leads to shrinkage in the size of memory bitcells, increasing its sensitivity to process variations introduced during manufacturing. Failure of a single bitcell can cause the failure of an entire memory; hence careful statistical analysis is essential in estimating the highest reliable performance of the bitcell before using them in memory design. With high repetitiveness of bitcell, the traditional method of Monte Carlo simulation would require a long time for accurate estimation of rare failure events. A more practical approach is importance sampling where more samples are collected from the failure region. Even though importance sampling is much faster than Monte Carlo simulations, it is still fairly time-consuming as it demands an iterative search making it impractical for large simulation sets. This thesis proposes two machine learning models that can be used in estimating the performance of a bitcell. The first model predicts the time taken by the bitcell for read or write operation. The second model predicts the minimum voltage required in maintaining the bitcell stability. The models were trained using the K-nearest neighbors algorithm and Gaussian process regression. Three sparse approximations were implemented in the time prediction model as a bigger dataset was available. The obtained results show that the models trained using Gaussian process regression were able to provide promising results.

Keywords: Bitcell, Performance, K-Nearest Neighbors, Gaussian Process Regression
Acknowledgments

I am incredibly grateful to my supervisor Hao Chi Kiang for the guidance and thorough explanations of the concepts whenever I needed. Your guidance and feedback were valuable during this thesis, and that made the entire work easier.

I would like to thank Mr Babak Mohammadi for giving me this opportunity and providing the data for this work. Thank you for taking out time in reviewing my work and providing valuable feedback.

My greatest gratitude to my industrial supervisor Tom Johansson for all the insightful suggestions, and for being available to answer all my questions throughout this project. All the discussions we had helped me in understanding the data and concepts. The inputs and ideas you provided helped me in planning and designing project.

A special thanks to my opponent Faton Rekathati and my examiner Anders Nordgaard for all the valuable feedback provided during the revision meeting.

I’d like to acknowledge the entire team at Xenergic for supporting me at various instances when I was in need.

I am particularly grateful to all students of Statistics and Machine Learning at Linköping University for making my master’s life memorable.

Finally, I would like to express the greatest appreciation to my family and Jones for the constant encouragement which was instrumental for me in the successful completion of this thesis.
# Contents

Abstract iii  
Acknowledgments iv  
Contents v  

## 1 Introduction  
1.1 Background ................................................. 1  
1.2 Aim ....................................................... 3  

## 2 Data  
2.1 Data preprocessing ........................................ 4  
2.2 Data description ......................................... 5  

## 3 Theory  
3.1 Supervised Learning ....................................... 7  
3.2 K-Nearest Neighbour ..................................... 8  
3.3 Bayesian Learning ....................................... 9  
3.4 Gaussian Process Regression ............................ 9  
3.5 Sparse Approximations for Gaussian Processes .... 12  

## 4 Method ................................................. 16  

## 5 Results  
5.1 Voltage prediction model ................................ 21  
5.2 Time prediction model .................................. 24  

## 6 Discussion  
6.1 Results .................................................. 31  
6.2 Method ................................................... 33  
6.3 Ethical considerations ................................... 34  

## 7 Conclusion ............................................. 35  

Bibliography ............................................... 36
1 Introduction

1.1 Background

Memory is a critical element for almost all modern Integrated Circuits (IC) [1]. Consider a processor which requires and generates a massive amount of data while in operation. This data can be stored and retrieved from a memory module either located outside or inside the chip. The memory units that are embedded within the chip are known as on-chip memories [2]. Compared to off-chip memories, on-chip memories have the advantages of faster access and less power consumption. Some of the on-chip memory types are SRAM, eDRAM, spin-transfer-torque magnetic RAM (STT-MRAM) and resistive RAM (RRAM) [3]. With an ever-increasing demand for performance and higher capacity, on-chip memory consumes most area in modern-day Integrated circuits (IC) [4]. Static Random Access Memory (SRAM) continues to be the predominantly used type of memory and is often used in applications to store data which needs faster access time [5]. The fundamental data storage module of an SRAM is a bitcell which is realized by a set of transistors, and it can store a single bit of data, i.e., logical ‘0’ or ‘1’. Two operations of a bitcell are read and write. Figure 1.1 shows the schematic diagram of a typical SRAM bitcell that uses six transistors (6T SRAM). Each bit is stored using four transistors marked M1 to M4. As shown in the diagram, the cell is connected to a wordline (WL) and a bitline (BL). The WL, along with the transistors M5 and M6, controls the access to the bitcell during read and write operation. The system uses the BL for either storing a desired value to the cell or reading the value already stored in it [6]. Depending on the requirements of the target application, the bitcells can be replicated to achieve the target memory size that can vary from a few kilobits to a few megabits.

ICs typically use a type of transistor called MOS - Metal Oxide Semiconductor to realize its functionalities and the process of manufacturing is known as semiconductor fabrication [7, p.220] [8, p.10]. In semiconductor industry, the manufacturing process is fast evolving, and each generation of the manufacturing process is known as a technology node which is expressed in units of length (micrometre or nanometre). For example, 16nm, 14nm, 7nm are some of the recent process technologies. It indicates a set of minimum dimensions for transistor and other features that can be achieved by a particular process technology [9] [10].
1.1. Background

Process technology advancements allow the shrinking of transistors’ dimensions. Smaller dimensions help to achieve higher transistor density to build ICs with improved performance and reduced energy consumption [10]. However, as transistor dimension scales down, process variations have become a severe issue in the design and fabrication of IC [11] [12] [13]. Process variations refer to the mismatch between the physical and electrical parameters of the transistor specified during design and the parameters after manufacture [14]. These variations such as random dopant fluctuations, line-edge roughness etc. [15, chapter 1] are introduced due to defect in the manufacturing process. The measurable impact is the deviation of circuit behaviour from its expected response affecting the yield of the electronic circuits. With the reduction in dimensions of transistors, accommodating the effect of process variations without affecting the reliability has emerged as a significant design challenge in circuit design [13] [15] [18]. Since, the bitcells, fundamental building blocks of SRAM, are shrunk more aggressively compared to other circuit components in the same technology, their sensitivity increase even further and makes them one of the first breakdown suspects for a failing IC.

The extreme sensitivity of the bitcell to process variations demands careful statistical methods in analyzing the failure probability before using them in memory design [13] [17]. The estimation follows a performance metric \( f(x) \) and a failure criteria \( f(x) > f_c \) where \( f_c \) is the minimum performance specification. Failure happens when a bitcell fails to meet the performance criteria during its operation. The failure probability is defined as \( P_f = P(f(x) > f_c) \) [19] [20]. In this thesis, two such performance criteria are considered for determining failure; they are:

1. **Time constraint**: Failure, in this case, is considered to have happened if the read or write operation does not complete in the constrained time [13].

2. **Minimum voltage needed to maintain bitcell stability**: In this case, the failure happens when the bitcell fails to maintain stability and flips the stored bit when voltage reduces beyond a threshold value [21].

---

https://en.wikipedia.org/wiki/Static_random-access_memory
A conventional statistical method for failure estimation is to run Monte Carlo simulations \[16\] \[17\]. Considering the fact that the state-of-the-art designs can have millions or more bit-cells with rare failure events arranged in a single SRAM, running Monte Carlo \[22\], p.123 would take long time for an accurate estimation which makes it computationally infeasible. A need for faster analysis has led to the adoption of more practical approaches such as importance sampling \[17\] \[16\] \[13\], which focus on collecting more samples from regions where the probability of failure is more. Given an exact distribution \( p(x) \), the samples are drawn from a proposal distribution \( q(x) \times f(x)p(x) \) \[23\]. The samples from proposal distribution are then used in failure probability estimation as:

\[
E[f] = \int f(x) \frac{p(x)}{q(x)} q(x) dx
\]  

where \( E[f] \) is the expected failure rate. However, without a proper choice of proposal distribution, the efficiency of importance sampling gets affected and identifying an optimal proposal distribution is challenging \[16\]. Also, there are multiple operational parameters which affect the bitcell. An iterative search is needed to find an optimal configuration of the operational parameters that can guarantee the failure rate within acceptable limits. Hence, even though importance sampling is much faster in comparison to Monte Carlo simulation, it is still fairly time-consuming.

Failure of a single bitcell can lead to failure of the memory \[16\]. Hence, estimating the highest reliable performance of a bitcell is critical in the design of a stable SRAM. Even though most of the existing approaches can guarantee accurate estimation, they are time-consuming to various extends. A machine learning model can be considered as a possible solution for faster performance estimation, which allows for models that can learn the behaviour of the bitcell from available observations. Once implemented, this model can predict the outcome of future events. A successful implementation that can give accurate predictions will result in a robust estimator which can overcome the delimitation of the existing methods by reducing the time needed for bitcell failure estimation. It would lead to a substantial reduction in the number of estimation cycles required for the in-house failure event estimator. Also, the model can be used for a quick overview of different input combinations to the bitcell during the design phase thereby providing a guidance to the designer.

1.2 Aim

The main objective of this thesis is to develop predictive regression models that could help in estimating the performance of a bitcell. Since an event of failure happens when a bitcell fails to meet the performance criteria, the machine learning models predict the values corresponding to these performance evaluation metrics. In this thesis, two models are trained where the first one predict the time required by the bitcell to complete its operation and the second model predict the voltage needed to maintain the stability of the bitcell. More distinctively, the research is based on the following questions:

1. Is it feasible to apply machine learning algorithms in predicting the performance of a bitcell?
2. How can the model be designed to predict the performance of a bitcell?
3. What would be the best model that can be fitted to the available data?
4. Are the predictions accurate enough to be used in bitcell performance estimation?
Data

Xenergic AB provided the data for this thesis. It consists of 80,879 data points with 30 features which include the simulated as well as the actual samples used in the design. For generating simulated dataset, random samples of input parameters were generated uniformly from their entire possible range, and they were passed through the in-house ‘rare failure event estimator’ to estimate the failure probability. As discussed in section [1], failure is decided based on the bitcell meeting predefined performance criteria. The available data contains the failure rate estimated based on two different performance metrics which are time constraint and minimum voltage to maintain bitcell stability. In the first case, failure happens when the bitcell fails to complete its operation in a constraint time. In the second case, the bitcell fails when the voltage drops below the threshold, causing the saved data in the bitcell to flip its content.

2.1 Data preprocessing

The efficiency of a machine learning model depends on the quality of the data used in training the model [24, chapter 3]. Several preprocessing techniques can be applied to the available data to improve its quality and in turn, the quality of predictions. Few examples include reducing noise, handling missing values, removing irrelevant or redundant variables and data transformations [24, chapter 3].

Two different models were trained in this work, and the target variable for the first model was time. For the second model, the target variable is the minimum voltage to maintain the stability of the bitcell. The data for both the models was available in a single dataset. The features [25, chapter 1.1] which were insignificant in training any of these models, as stated by the domain experts, were discarded [24, chapter 3]. Also, features with more than 90% of missing values were discarded as no imputation techniques [26] are valid in this case. In the available dataset, close to 70% of the samples have a failure probability of either 0 or 1. The values of interest in this work are extremely small failure probabilities associated with rare failure events. Failure probability of 1 is associated with a bitcell that fails in all cases, and in this thesis work, we are not aiming at estimating those values; hence corresponding samples were dropped. Failure probability of 0 corresponds to correctly working cases which
again are not of interest as the aim is to predict rare failure events, hence these samples were dropped as well. After cleaning the dataset, a total of 23,528 samples were available for training the prediction models, in which 19,274 uses time constraint, and 4254 uses voltage for maintaining stability as the performance metric. The categorical variable ‘simulation type’ described in table 2.1 is used in capturing the performance metric considered in each of the available samples.

In regression [27, chapter 1] analysis, ‘dummy variables’ are used to encode information contained in features that are not measured on numerical scales [28]. They are numerical variables used for representing categorical data. For a categorical variable with ‘k’ different categories, k dummy variables are created, one for each category and value of one is assigned if the observation belongs to that category and 0 otherwise. For each of the three categorical features listed in table 2.1, dummy variables were created and assigned a value of 1 for the observations belonging to that category.

The measurement unit of each feature in data can affect data analysis [24]. Normalization is a useful preprocessing technique that can be used in this case to give equal weights to all the features. The features in the available dataset were measured in different scales, and so they were normalized by removing the mean and scaling to unit variance using the formula

\[ x' = \frac{x - \mu}{\sigma}, \]

where \( x' \) is the standardized feature value, \( \mu \) the mean and \( \sigma \) the standard deviation. In this thesis, to train the regression models, KNN and the Gaussian process regression are used, and both of them are distance-based algorithms. The normalization here helps to prevent the features with large range from outweighing the features with smaller ranges [24].

### 2.2 Data description

Table 2.1 and 2.2 lists the categorical and numerical variables, respectively.

**Table 2.1: Categorical variables used in the machine learning models and their description**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitcell</td>
<td>This variable provides information about the manufacturer as well as some of the manufacturing specifications associated to the bitcell</td>
</tr>
<tr>
<td>Simulation type</td>
<td>This represents the metrics used to measure the failure rate of a bitcell.</td>
</tr>
<tr>
<td>Corner</td>
<td>Process corners represent the extremes of parameter variations that affects a bitcell.</td>
</tr>
</tbody>
</table>
### Table 2.2: Numerical variables used in the machine learning models and their description

<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{dd}$</td>
<td>Voltage supply to a bitcell</td>
</tr>
<tr>
<td>$V_{bl}$</td>
<td>Voltage level on a bitline</td>
</tr>
<tr>
<td>$V_{wl}$</td>
<td>Voltage supplied to the wordline of SRAM</td>
</tr>
<tr>
<td>Time constraint</td>
<td>The time duration for a successful read or write operation for a given voltage level. Read or write is considered unsuccessful if it does not complete in constrained time.</td>
</tr>
<tr>
<td>$C_{bl}$</td>
<td>Parasitic capacitance on the bitline</td>
</tr>
<tr>
<td>$V_{nbl}$</td>
<td>Nbl stands for Negative bitline. Vnbl is the negative voltage applied to the lower bitline to help writing.</td>
</tr>
<tr>
<td>Read bitline offset</td>
<td>The difference in voltage required between the bitlines to consider a read operation to be successful.</td>
</tr>
<tr>
<td>Write node offset</td>
<td>The difference in voltage that is required between the nodes of the bitcell to consider a write operation to be successful.</td>
</tr>
<tr>
<td>$V_{bpmos}$</td>
<td>The voltage for the body of the PMOS transistors. These are the transistors marked M2 and M4.</td>
</tr>
<tr>
<td>Temperature</td>
<td>The physical temperature of the operating environment.</td>
</tr>
<tr>
<td>Failure probability</td>
<td>The rate of failure of a bitcell.</td>
</tr>
</tbody>
</table>
This chapter gives a detailed account of various machine learning algorithms that are used in this work.

3.1 Supervised Learning

The goal of supervised learning algorithms is to learn the mapping from input $x$ to output $y$ where $x \in \mathbb{R}^N$ \cite{chapter1}. In regression problems, the output $y$ is continuous and $y \in \mathbb{R}$ \cite{chapter1}. This problem can be formalized as function approximation where an unknown function $y = f(x)$ is assumed with the goal of approximating $f$. Regression can be considered as a task in function approximation where quantitative outputs are predicted. Training is done using the available data and the function is estimated. The optimal choice of function $f(x)$ is the one which minimizes the expected loss which is given by:

$$E[L] = \int \int L(y, f(x)) p(x, y) \, dx \, dy$$

This function is used in making predictions $\hat{y} = \hat{f}(x)$. The training data can be both quantitative and qualitative. The main goal is to generalize by making predictions on novel data that consists of previously unseen inputs \cite{chapter1} \cite{chapter2} \cite{chapter1}.

Probabilistic models of the form $p(y|x)$ are defined in many ways, and an important distinction is based on the number of parameters used in the model. Models having a fixed number of parameters are known as parametric models. When the count of parameters grows with the amount of training data, they are non-parametric models. Parametric models are faster, simple to implement and easy to interpret, but they make strong assumptions about the underlying data distribution. In contrast, non-parametric models are more flexible and they make very mild structural assumptions, but its computational complexity increases with the size of the dataset \cite{chapter2} \cite{chapter1}.
3.2 K-Nearest Neighbour

In pattern recognition, the most simple non-parametric function approximation is the K-nearest neighbour (KNN) algorithm which was formalized by Cover and Hart in 1967 \[30\]. KNN is a memory-based learning algorithm used when little or no prior knowledge about the underlying data distribution is known. The observations in the training set that are closest in the input space to the target are used in prediction. Suppose the training data set has \(N\) samples and each input sample \(x_i\) is defined by \(m\) features and an output variable \(y_i\) as \(x_i = \{(x_{i1}, x_{i2},..,x_{im}, y_i)\}\). The KNN algorithm learns the function \(y = f(x)\) and the output is computed as follows:

\[
\hat{y} = \frac{1}{K} \sum_{i \in N_k(x,D)} y_i
\]

where \(y_i\) is the output values of K neighbouring points to the target and \(N_k(x,D)\) are the indices of K nearest points to \(x\) in the training dataset \(D\) \[27\] p.17. Value of parameter \(K\) should be optimum for the algorithm to achieve better performance. A large value can smooth out the impact of noise in training samples, whereas a small value reduces the algorithm complexity \[31\] p.235.

To make predictions using KNN, a metric should be defined to calculate the distance between the data on which a prediction is to be made and the training data in feature space. The most popular choice of distance metric is the Euclidean distance which is given by:

\[
d_{x,y} = \sqrt{\sum_{j=1}^{f} (x_j - y_j)^2}
\]

KNN works as follows \[31\] p.232:

- Set the value of \(K\), the selected number of neighbors
- Given a query instance \(x_q\) whose output value is to be predicted, select \(x_1, x_2,..,x_k\) which are the \(K\) nearest neighbours to the query instance \(x_q\), where \(x_i = (x_{i1}, x_{i2}...x_{im}, y_i)\)
- Return the mean of the labels of \(K\) neighbors

\[
\hat{y} = \frac{1}{K} \sum_{i=1}^{K} y_i
\]

KNN is simple and efficient in many practical applications. It is robust to noisy training samples when a large dataset is available. The disadvantage of this method is that it needs the entire training set to be stored. Also, distance of test data with all the training set needs to be calculated which leads to need for expensive computation as training samples increases \[29\] p.127.
3.3 Bayesian Learning

In Bayesian learning [27, chapter 3.2] approach, the problem is provided with a probabilistic treatment to predict the output along with uncertainties. Assume \( N \) input data points \( D = (x_1, y_1), (x_2, y_2), ..., (x_N, y_N) \) where \( x_i \in \mathbb{R}^m \). The task here is the inference of the distribution parameters \( \theta \) given the observed data \( D \). The likelihood of data which is assumed to be independent and identically distributed given the parameter is defined as:

\[
P(D|\theta) = \prod_{i=1}^{N} p(x_i|\theta)
\]

Bayesian inference uses Bayes theorem, where a prior knowledge about the parameters \( P(\theta) \) is combined with the likelihood to get the posterior distribution of the parameters. [31, chapter 8]

\[
p(\theta|D) = \frac{P(D|\theta) \times P(\theta)}{p(D)}
\]

3.4 Gaussian Process Regression

Rasmussen and Williams define a Gaussian Process (GP) as "a collection of random variables, any finite number of which have (consistent) joint Gaussian distributions" [32, chapter 2]. Gaussian process regression follows a Bayesian approach to estimate \( p(y|X) \) with input data \( X = [x_1, ..., x_N] \) and observed output \( y = [y_1, ..., y_N] \). The random variables are represented by a set of latent functions \( f = [f_1, ..., f_N] \) where \( f(x) \) is the function values evaluated at any input point \( x \). The prior belief over this latent variables is defined using a GP [32, chapter 2]

\[
f(x) \sim GP(m(x), k(x, x'))
\]

with prior mean,

\[
m(x) = E[f(x)],
\]

and covariance function

\[
k(x, x') = E[(f(x) - m(x))(f(x') - m(x'))]
\]

The kernel (covariance) function in equation (3.2) is a positive definite function of two inputs \( x \) and \( x' \) defining the prior covariance between the latent function values. Supervised learning methods assume that the points closer in input space will correspond to similar target values. Covariance functions [32, chapter 4] are used in defining this similarity in the Gaussian process regression. Many different covariance functions can be chosen for modelling the GP as each of them correspond to different types of assumptions about the distribution of the modelled function. Also, each of them has a set of hyperparameters which are used in specifying the shape of the function. A common choice is the squared-exponential (SE) covariance function also known as radial basis functions (RBFs) which is of the form

\[
k(x, x') = \exp \left( -\frac{(x-x')^2}{2l^2} \right)
\]

where \( l \) is the length scale parameter which defines the width of the kernel. It assumes that the modeled function can have derivatives of all orders and is thus very smooth. To model functions where the strong smoothness assumption is impractical, another variant of the kernel function called the Matérn class of covariance functions can be used. They are of the form

\[
k(x, x') = \frac{2^{1-v}}{\Gamma(v)} \left( \frac{\sqrt{2v|x-x'|}}{l} \right)^v K_v \left( \frac{\sqrt{2v|x-x'|}}{l} \right),
\]
3.4. Gaussian Process Regression

with lengthscale parameter $l$. $K_v$ is a modified Bessel function \cite{33} p.356, 357 of order $v$, $v > 0$. This kernel assumes that the function is $[v]$ times differentiable and hence the smoothness of the kernel is flexible depending on the value of parameter $v$. Both SE and Matérn covariance functions are stationary which means that they are invariant to translations and the output depend only on the difference \[ |x - x'| \]. \cite{32} \cite{34}. In cases where the GP model produce different predictions when the inputs $x$ are moved, non stationary covariance functions can be used. A Linear kernel is an example of non stationary covariance function \cite{32} chapter 4 and is of the form

$$k(x, x') = (x - c).(x' - c)$$

where $c$ is the offset determining the x-coordinate of point through which all lines in the posterior passes \cite{34}. For specific applications which require complex kernels, new kernels can be built out of simpler kernels by addition or multiplication. The requirement of the newly constructed kernel $k(x, x')$ is that it should be symmetric and positive definite and it should be able to express the appropriate form of similarity between $x$ and $x'$ for the specific application or dataset \cite{29} p.296.

For inference using a GP model, a joint posterior on function values $f$ at training inputs and $f_*$ at test points $X_*$ is obtained by combining the prior on functions with likelihood $p(y|f)$ using the observed training data. Using Bayes rule, the joint posterior is derived as \cite{35}

$$p(f, f_*|y) = \frac{p(f, f_*|y)}{p(y)}$$

The posterior predictive distribution over the targets is derived by marginalizing out the latent functions of the training set.

$$p(f_*|y) = \int p(f, f_*|y)df = \frac{1}{p(y)} \int p(y|f)p(f, f_*)df$$

So, with $K(X, X)$ as the covariance matrix of the training inputs, $K(X_*, X_*)$ the covariance matrix of the test inputs, $K(X_*, X)$ the covariance matrix between training and test inputs and $\sigma_n^2$ as the noise variance, the Gaussian predictive distribution is obtained as:

$$f_*|X, y, X_* \sim \mathcal{N}(\tilde{f}_*, \text{cov}(f_*))$$

where,

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

$$\text{cov}(f_*) = K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1}K(X, X_*) \quad (3.3)$$

The posterior GP depends on the parameters of the covariance function $\theta$, and the noise variance $\sigma_n^2$. These can be estimated by maximizing the log marginal likelihood.

From the above equations, the mean prediction is a linear combination of observed values $y$. Also, it is dependent only on the training inputs and the correlation of this particular test point to the training inputs. The predictive variance depends only on the observed inputs and not the targets. It is the difference between the prior covariance and the information provided by the training data \cite{27} \cite{35} \cite{32}.
Optimizing hyperparameters

In a GP model, the prior covariance function \( K = k(x, x') \) is parameterized by a vector \( \theta \) known as hyperparameters that specify the shape of the covariance function. These are evaluated at the training data points and joint likelihood is written as [36]

\[
p(y, f|X, \theta) = p(f|X, \theta) \prod_{n=1}^{N} p(y_n|f_n)
\]

where \( p(y_n|f_n) \) is the noise model. When the model is Gaussian, the marginalized likelihood is computed by integrating the joint likelihood over \( f \) and as [32] chapter 2, 3

\[
p(y) = \int p(f|X, \theta) \prod_{n=1}^{N} p(y_n|f_n) df
\]

The hyperparameters \( \theta \) can be optimized by maximizing the marginal likelihood calculated in equation (3.4) with respect to these parameters.

Assumed Density Filtering for Gaussian Process models

In cases where the noise model is non Gaussian, the marginalization is intractable. A possible approximation here is Assumed Density Filtering (ADF) suggested Csató, 2002 [37]. In ADF scheme, each factor of noise model is approximated using a Gaussian distribution \( \mathcal{N}(z_n|f_n, \beta_n^{-1}) \) with mean \( z_n \) and variance \( \beta_n^{-1} \). The approximation to the true posterior process \( p(f|y) \) is build by adding each factor at a time such that after \( i \) points gets added, an approximation \( q_i(f) \) is obtained. An initial prior on the GP \( q_0(f|X, \theta) = p(f|X, \theta) \) is assumed and an approximation to posterior distribution is build by selecting a datapoint \( n_1 \) to get the posterior of the form \( \hat{p}_1(f) \propto q_0(f)p(y_n|x_n) \). Then the KL divergence between \( \hat{p}_1(f) \) and a Gaussian distribution \( q_1(f) \) is minimized to get the updated posterior approximation [36].

\[
KL(\hat{p}_1(f)|q_1(f)) = \int \hat{p}_1(f) \ln \frac{\hat{p}_1(f)}{q_1(f)} df
\]

The approximation is done repeatedly until all data points are included. To be more generic, when the \( i \)th data point is included, the posterior is of the form

\[
\hat{p}_i(f) = \frac{q_{i-1}(f)p(y_n|x_n)}{Z_i}
\]

where

\[
Z_i = \int p(y_n|f_n)q_{i-1}(f) df
\]

is the normalization constant. Now, the KL divergence between \( KL(\hat{p}_i(f)|q_i(f)) \) is minimized to get the posterior as

\[
q_i(f) = \mathcal{N}(f|\mu_i, \Sigma_i)
\]

Since this is a sequential process, the mean \( \mu_i \) and covariance matrix \( \Sigma_i \) also has to be updated iteratively. According to ADF approximation scheme, the following update equations hold. More detailed derivation can be seen in Lawrence et al., 2004 [38] [36].

\[
\mu_i = \mu_{i-1} + \Sigma_{i-1} g_i
\]

\[
\Sigma_i = \Sigma_{i-1} - \Sigma_{i-1} (g_i g_i^T - 2\Gamma_i) \Sigma_{i-1}
\]

where \( g_i = \frac{\partial \log Z_i}{\partial \mu_i} \) and \( \Gamma_i = \frac{\partial \log Z_i}{\partial \Sigma_i} \).
If the noise model is Gaussian, the above approximation can be applied as follows. The matrix \((g_i g_i^T - 2 \Gamma_i)\) which was considered for updating covariance matrix can be written in diagonal form with each diagonal element given by

\[ v_{in} = \frac{1}{\bar{\zeta}_{(i-1,n)} \bar{\beta}_n^{-1}} \]

(3.5)

where \(\bar{\beta}_n^{-1}\) is the variance of the noise model and \(\bar{\zeta}_{(i-1,n)}\) is the nth diagonal element of \(\Sigma_{i-1}\). \(\Sigma_{i-1}\) is the covariance matrix with \(i - 1\) included data points. All other elements of the matrix are considered as 0. The mean and variance update equations gets simplified as

\[ \mu_i = \mu_{i-1} + g_{in} s_{i-1,n} \]
\[ \Sigma_i = \Sigma_{i-1} - v_{in} s_{i-1,n} s_{i-1,n}^T \]

where \(s_{i-1,n}\) is the nth row of \(\Sigma_{i-1}\) [38].

Hence this ADF approximation scheme can be easily applied to any noise model where the expectation of noise model is tractable [36] [38].

3.5 Sparse Approximations for Gaussian Processes

A Gaussian process approach involves inversion of a matrix of size \(N \times N\) where \(N\) is the size of the training dataset, and it scales as \(O(N^3)\). Different sparse methods and approximations are discussed in the literature to make scalable GP where only a set of latent variables are given an exact approximation, and the rest of the variables are approximated using computationally cheaper methods [35].

Most of the sparse algorithms in literature construct an approximation for the Gaussian process using a set of inducing points. The inducing variables are the function values at some of the training data points. Inducing inputs are identified, and the hyperparameter learning of the posterior GP happens by minimizing the log marginal likelihood. Many methods in literature also concentrate on getting an approximation to the marginal likelihood using the inducing points. The new set of selected data points is referred to as the active set. The selection of inducing points helps in making the GP scalable and affordable considering the computations [39].

Rasmussen and Williams [32, chapter 8] discuss the sparse approximation algorithms in literature and few among them are discussed here. Silverman, 1985 [40] showed that a mean GP predictor can be obtained from a finite dimensional linear regression model as:

\[ f(x_*) = \sum_{i=1}^{n} a_i k(x_*, x_i) \]

with prior alpha \(a = \mathcal{N}(0, K^{-1})\), \(K\) is the covariance matrix. Wahba proposed a simple approximation to the above model, 1990 [41, chapter 7] where only a subset of the regressors (SOR) is used in approximation.

\[ f_{SR} = \sum_{i=1}^{m} a_i k(x_*, x_i) \]

with \(a_m = \mathcal{N}(0, K_{mm}^{-1})\). The above method has time complexity \(O(nm^2)\), and it also takes \(O(m)\) for prediction of mean and \(O(nm)\) for predictive covariance. Rasmussen and Quiñonero-Candela, 2005 [42] suggested a variation to the SOR method where the covariance matrix inversions can be partitioned and implemented. The most simple method to
approximate a GP is selecting a subset of data points \( m < n \) which reduces the computational complexity of the algorithm as \( \mathcal{O}(m^3) \) \[\text{[32, chapter 8]}\]. Seo et al., 2000 \[\text{[43]}\] suggested an active selection strategy based on the posterior variance, where the selected set includes data points that exhibit the maximum predictive variance. Herbrich et al., 2003 \[\text{[44]}\] proposed a greedy subset selection algorithm by using the information gain score. Csato and Opper, 2002 \[\text{[45]}\] used a bayesian online algorithm where the subset of data is constructed sequentially.

**Greedy Approximation**

It is assumed that, selection of an optimal subset of size \( m < n \) is impossible \[\text{[32, chapter 8]}\]. A simple approach to obtain an active set is random selection. Studies have shown that, on considering combined statistical and computational efficiency, random subset selection leads to poor results compared to a greedy procedure that follows some criterion for active set selection \[\text{[38]}\].

**Algorithm 1: General subset selection framework \[\text{[32, chapter 8]}\]**

| Input: \( m \), desired size of active set |
| Output: \( I \), active set |
| Initialization: \( I = \phi \), \( R = \{1, ..., n\} \) |
| for \( j \leftarrow 1 \) to \( m \) do |
| Create working set \( J \subseteq R \) |
| Compute \( \Delta_j \) for all \( j \in J \) |
| \( i = \arg\max_{j \in J} \Delta_j \) |
| Update the model to include data from example \( i \) |
| \( I \leftarrow I \cup \{i\} \) |
| \( R \leftarrow R / \{i\} \) |
| end for |

Algorithm 1 is a general framework for greedy subset selection suggested by Rasmussen and Williams, 2006 \[\text{[32, chapter 8]}\]. The algorithm starts with an empty active set \( I \) and training set \( R \) where a data index is iteratively selected from \( R \) and is added to \( I \) based on some criterion \( \Delta \). Selection of the active point from the entire training set by running optimization can be expensive based on the size of the training set. In these scenarios, a subset \( J \) of the training set \( R \) is selected for optimization. Using this greedy approximation, subsets can be selected using any valid selection criteria.

**Active Learning Method**

Active learning is a field of Machine learning where the learning algorithm chooses the data from which it learns \[\text{[46]}\]. Unlike usual regression problems where the learner is supplied with data in advance, here the learner chooses the data to incorporate more information. A weaker model of active learning is query selection where the learner selects the queries from a stream of data points. Seo et al., 2000 suggested criteria for active data selection for the GP. Assuming that the given covariance function and noise model for GP is correct, this method focuses on selecting the data points that exhibit maximum predictive variance. In the Gaussian process regression, the predictive variance of query data points can be obtained easily from equation \[\text{[5.3]}\]. Hence, using the generalized framework for data selection given in algorithm 1, these criteria can be used to choose the most promising query candidates from the training dataset. This query strategy depends on the covariance function and the input data present and does not dependent on the target values of training data \[\text{[43]}\].
Informative Vector Machine

To get a fast greedy algorithm for obtaining a sparse representation of data, Lawrence et al., 2003 [44] suggested the Informative Vector Machine (IVM) algorithm.

Algorithm 2: The IVM algorithm [38]

Input: \( m \), desired size of active set
Output: \( I \), active set
Initialization: \( \varsigma_0 = \text{diag}(K), J = \{1, \ldots, n\}, I = \{\} \), \( M_0 \) is an empty matrix

for \( 1 \leftarrow 1 \) to \( m \) do
  for all \( n \in J \) do
    compute \( \nu_{in} \) according to 3.5
    compute \( \Delta H_{in} \) according to 3.6
  end for
  \( n_i = \arg \max_{n \in J} \Delta H_{in} \)
  compute \( \varsigma_i \) using 3.9
  Append \( \nu_{in}, \varsigma_{i-1,ni} \) to \( M_{i-1} \) to form \( M_i \)
  Add \( n_i \) to \( I \) and remove \( n_i \) from \( J \)
end for

Algorithm 2 gives the detailed implementation steps. This gives an approximation with complexity \( O(m^2N) \) where \( m \) is the number of data-points included in the active set \( I \). The selection heuristic in this algorithm is designed based on active learning strategy. The criteria for greedy selection of data points in this method is differential entropy score based on Information theory. The entropy change on addition of a data point is given by the equation:

\[
\Delta H_{in} = -\frac{1}{2} \log(1 - \nu_{in}\varsigma_{i-1,ni})
\] (3.6)

where \( \nu_{in} \) is defined by equation (3.5) and \( \varsigma_{i-1,ni} \) is the nth diagonal element of \( \Sigma_{i-1} \). IVM includes data points in a sequential manner such that the addition of those data points minimizes the entropy of the posterior process. From equation (3.6), calculation of the change in entropy using \( \nu_{in} \) and \( \varsigma_{i-1,ni} \) requires storing of the entire covariance matrix. Instead of storing the entire posterior covariance matrix in memory, the algorithm uses structure in equation (3.7) to derive the current covariance matrix by adding successive outer products to the original prior covariance \( K = \Sigma_0 \).

\[
\Sigma_i = K - M_i^T M
\] (3.7)

where \( K \) is the prior covariance matrix \( \Sigma_0 \) and kth row of \( M \) is \( \sqrt{n_k}k_{nk}\). \( n_k \) is the kth included data point. \( s_{i-1,ni} \) can also be derived as

\[
s_{i-1,ni} = k_{ni}a_{i-1,ni}^TM_{i-1}
\] (3.8)

where \( k_{ni} \) is the \( n_i \)th row of the prior covariance matrix and \( a_{i-1,ni} \) is the \( n_i \)th column of \( M_{i-1} \). The diagonal of covariance matrix can be updated using the equation

\[
\varsigma_i = \varsigma_{i-1} - \nu_{in}d\text{diag}(s_{i-1,ni}s_{i-1,ni}^T)
\] (3.9)

The computational complexity of this algorithm is \( O((i-1)N) \) which in turn leads to a sparse approximation of the GP with complexity \( O(m^2N) \) [38]. The algorithm is designed to compute the values required during the process, which makes it an efficient approximation model for GP.
Variational Learning of Inducing Variables

Titsias, 2009 [39] formulated a sparse method which directly approximates the mean and covariance of the posterior GP using a small set of \( m \) inducing inputs \( X_m \) and latent function values \( f_m \) evaluated at the inducing inputs. In Gaussian process regression, the posterior distribution \( p(f|y) \) is approximated using the latent variables \( f \). In this method, a GP prior can be augmented as

\[
p(f, f_m) = p(f|f_m)p(f_m).
\]

If \( f_m \) is a sufficient statistic for \( f \), then output \( y \) and \( f \) are conditionally independent of \( f_m \) then the true posterior can be approximated as

\[
p(f, f_m|y) = p(f|f_m)p(f_m|y).
\]

Finding a sufficient statistic is difficult in a real scenario and hence \( \phi(f_m) \neq p(f_m|y) \) is chosen to be a free variational Gaussian distribution and an approximate variational posterior is defined as

\[
q(f, f_m) = p(f|f_m)\phi(f_m)
\]

The function \( \phi \) and the inducing inputs \( X_m \) are jointly selected by minimizing the distance between augmented true posterior \( p(f, f_m|y) \) and augmented variational posterior using KL divergence \( KL(q(f, f_m)||p(f, f_m|y)) \). The underlying optimal distribution for sparse representation is reached by tuning these variational parameters [39].
The first step of the research was to analyze the data and propose a model. A detailed study was done to decide on the modelling strategy. The initial phase of the research focused on predicting the failure probability. On further analysis, a different approach where the failure probability act as an input feature to the model was proposed. As discussed in section 1.1, failure is assessed based on bitcell meeting certain performance criteria during its operation. The two main causes of failure are 1) the bitcell taking an increased time in read or write operation and 2) bitcell fails to maintain stability at a given voltage and the value stored in the cell gets flipped. Two different prediction models were proposed based on the above metrics.

The performance criteria considered for the first model is time. The aim of statistical analysis is to determine if the failure rate remains within a limit. So, an acceptable failure probability for the bitcell is considered as a covariate along with other operational features. The model predicts the time required for a bitcell to complete an operation with this specified failure rate.

The second measurement metrics for failure estimation is the minimum voltage required to maintain the stability of bitcell so that the data stored in the cell does not flip. The model here takes the failure probability along with other relevant operational features as input and predict the minimum voltage needed to maintain bitcell stability.

Since the target variables are continuous values, regression methods has to be used in making predictions. The first algorithm implemented to predict the performance of a bitcell is the KNN which is a simple non-parametric algorithm for regression as well as classification problems. In this thesis, KNN is considered as the benchmark implementation. The second algorithm implemented is the Gaussian process (GP) regression. A GP is a flexible non-parametric method to deal with non-linear regression problem and provides a probabilistic formulation of the model [45]. The GP was approximated using three different sparse approximation techniques in the time prediction model to overcome the computational complexity of training the GP using a dataset with 19,274 data points.
**Exploratory data analysis**

To quantitatively understand the available data and its characteristics, the dataset was analyzed using various visualization techniques [47]. Figure 4.1 shows the actual distribution of the target variable of the first model; time. The duration of time is short and bounded, and so the plot doesn’t give any useful information, so time values were log-transformed. Figure 4.2 shows an approximate log-normal distribution of time with few outliers. Figure 4.3 shows the distribution of voltage, the target variable of the second model.

![Figure 4.1: Frequency distribution of target variable time](image1)

![Figure 4.2: Frequency distribution of target variable time - log transformed](image2)

![Figure 4.3: Frequency distribution of target variable voltage](image3)

Correlation analysis [24, chapter 3] helps in recognizing the relationships and patterns in data and also in identifying the significance of features in predicting the target variable. Table 4.1 presents the correlation of all numerical features to the target variable time. The variables $V_{dd}$, $V_{bl}$, $V_{wl}$, $V_{bpmos}$, Write node offset, and temperature show a significant negative correlation to the target value, whereas $N_{bl}$ shows considerable positive correlation. Table 4.2 presents the correlation of numerical predictor variables to the minimum voltage.
to maintain bitcell stability. The variables Vbl and Vwl show a high correlation to the target, which implies that they will have a significant impact in making the predictions.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Correlation to time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{dd}$</td>
<td>-0.59</td>
</tr>
<tr>
<td>Vbl</td>
<td>-0.59</td>
</tr>
<tr>
<td>Vwl</td>
<td>-0.52</td>
</tr>
<tr>
<td>Cbl</td>
<td>-0.04</td>
</tr>
<tr>
<td>Nbl</td>
<td>0.42</td>
</tr>
<tr>
<td>Read bitline offset</td>
<td>0.19</td>
</tr>
<tr>
<td>Write node offset</td>
<td>-0.30</td>
</tr>
<tr>
<td>Failure probability</td>
<td>0.17</td>
</tr>
<tr>
<td>Temperature</td>
<td>-0.58</td>
</tr>
<tr>
<td>Vbpmos</td>
<td>-0.35</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable</th>
<th>Correlation to voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vbl</td>
<td>0.87</td>
</tr>
<tr>
<td>Vwl</td>
<td>0.93</td>
</tr>
<tr>
<td>Failure probability</td>
<td>-0.30</td>
</tr>
<tr>
<td>Temperature</td>
<td>-0.01</td>
</tr>
<tr>
<td>Vbpmos</td>
<td>0.48</td>
</tr>
</tbody>
</table>

**Hyperparameter Optimization**

For any algorithm, the selection of hyperparameters has a significant impact on deciding the quality and performance of the model. The most crucial step in the K-nearest neighbour algorithm is the selection of the optimal value of ‘K’. An elbow plot determination was used in deciding the optimal ‘K’. Elbow method is an optimization heuristic in mathematics. Here the error is plotted as a function of the number of neighbours. The elbow of the curve is chosen as the optimal number of neighbours as the change in error after that point is trivial or it starts increasing.

In case of Gaussian processes, the selected covariance function mainly defines a GP [32, chapter 5] and choosing optimal hyperparameters have a crucial impact in determining the efficiency of the trained model. This research used SE, Matérn and Linear kernels. For all the three covariance functions, the signal variance is the hyper-parameter to be optimized with
an additional lengthscale parameter for SE and Matérn kernels. The Gaussian noise variance is not a kernel parameter, but the Gaussian likelihood is considered in evaluating the kernel function; hence Gaussian noise variance is also optimized [32, chapter 5]. In this thesis, the function ‘optimize’ in package Gpy [48] is used for optimizing the hyperparameters. This function uses the L-BFGS-B optimization algorithm in selecting the hyperparameters by minimizing the negative log marginal likelihood.

**Evaluation Metrics**

Evaluation using proper metrics is critical in regression modelling as it is necessary to build a good quality model. Two statistical measures were used for assessing the goodness of fit.

**Root Mean Squared Error** [49] is an evaluation metric for quantitative measurement of the model performance. It is the average squared difference between the actual value and the predicted value. RMSE is useful in comparing different models as well as for optimizing the model hyper-parameters.

\[
RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y})^2}
\]

**Explained variance score** measures the proportion to which the given model explains the variation in the provided dataset [49].

\[
explained\_variance(y, \hat{y}) = 1 - \frac{Var(y - \hat{y})}{Var(y)}
\]

where \( y \) is the actual value, \( \hat{y} \) is the predicted value and \( Var \) is the variance. The highest possible explained variance score of 1.0 means that the model is able to capture the complete variance in the input data.

**Graphical Analysis of Residuals**

Residual refers to the difference between the actual value and the predicted value. In statistical modelling, examining the residual is substantial in determining the goodness of fit. Histogram of residuals, normal probability plot and residual plots are few among various visualization techniques to analyze the residual distribution. A normal probability plot is a technique for analyzing if the given data deviates from the normal distribution [51]. In this, the residuals are plotted against a theoretical normal distribution. If the points follow an approximately straight line, then the residuals are assumed to be normally distributed and if it deviates from the straight line, it indicates the deviation from normality. Histogram of residuals [50] is another standard graphical visualization method used in checking if the residuals are normally distributed. Another technique is to use a residual plot which is obtained by plotting predicted values against the residuals. Using this plot, one can check for the remaining trends in data that are not captured by the model. A residual plot can also be used to check if the variance of the residuals is constant. Randomly distributed residuals around the horizontal axis is an indication of a good fit. The occurrence of patterns in the residual plot suggests a better non-linear model in fitting the data [52].

**Model selection**

In machine learning regression problems, a generalized model should be trained that can perform well on previously unseen data. One way to test the generalization performance of the model is by dividing the dataset to train, test and validation sets and using them in evaluation [27, chapter 1]. The dataset was divided in 80:20 ratio with 80% of the data for the training set and 20% for the test set. 20% of the training dataset was randomly selected to
use as the validation set. The training dataset is used for training the model. The validation dataset evaluates the performance of the model fit on the training dataset with different hyperparameter values \[27\] chapter 1. Test dataset provides an unbiased evaluation by providing an estimate of how well the final model can generalize to new and unseen data. The model with the best generalization performance is selected as the final model \[27\] chapter 1. One disadvantage of this method is the sensitivity of the performance evaluation to the way data set is partitioned into training, validation and test set.
This chapter has two parts. The first part presents the results for the voltage prediction model and the second part presents the results for the time prediction model.

5.1 Voltage prediction model

For training the model, 4254 data points were available, and it was divided into train, test and validation sets.

K-nearest neighbours

K-nearest neighbours algorithm was used to train the model for predicting the minimum voltage required to maintain the stability of the bitcell.

Figure 5.1: RMSE of train and validation data for different values of K
5.1. Voltage prediction model

RMSE against K for both train and validation sets was plotted, as shown in figure 5.1, to choose the optimal number of neighbors. From the plot, it can be seen that the training error is least at lower values of K but not the validation errors. This is happening because the model is overfitting the train data for lower values of K. The validation error has the lowest value at five; hence the final model used K = 5. It was applied on test dataset to check the generalization performance. The model was able to make predictions with RMSE score of 0.05 and an explained variance score of 0.934.

Gaussian process regression

Gaussian process regression models using different kernels are trained on the dataset. The predictions on new test points are made by considering the joint distribution of the target values and the function values at the test points. Kernel hyperparameters were optimized before making predictions. For each test point, the model outputs the mean and the variance of the predictive posterior distribution.

Table 5.1: RMSE and explained variance score of voltage prediction models using various kernels

<table>
<thead>
<tr>
<th>Kernel</th>
<th>RMSE</th>
<th>Explained Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>0.041</td>
<td>0.971</td>
</tr>
<tr>
<td>SE</td>
<td>0.04</td>
<td>0.981</td>
</tr>
<tr>
<td>Matérn(_{v=3/2})</td>
<td>0.021</td>
<td>0.988</td>
</tr>
<tr>
<td>SE + Linear</td>
<td>0.027</td>
<td>0.991</td>
</tr>
<tr>
<td>SE * Linear</td>
<td>0.034</td>
<td>0.982</td>
</tr>
<tr>
<td>Matérn(_{v=3/2}) + Linear</td>
<td>0.023</td>
<td>0.993</td>
</tr>
<tr>
<td>Matérn(_{v=3/2}) * Linear</td>
<td>0.031</td>
<td>0.984</td>
</tr>
<tr>
<td>SE + Matérn(_{v=3/2})</td>
<td>0.025</td>
<td>0.991</td>
</tr>
</tbody>
</table>

Table 5.1 shows the RMSE and explained variance score of the predictions on validation dataset. From the scores, it is noticed that the GP model trained using a linear kernel can make predictions close to the GP model trained using other complex kernels. The model trained using Matérn\(_{v=3/2}\) kernel gives the predictions with least RMSE score of 0.021 and the additive combination of linear and Matérn\(_{v=3/2}\) kernels give the highest explained variance score. Since Matérn\(_{v=3/2}\) kernel is simple compared to the composite kernels and the difference in the explained variance is not substantial, it was chosen to be the final kernel to train the GP regression model.

In the GP regression model, the residuals are assumed to have i.i.d Gaussian distribution\([32]\); hence they are graphically analyzed to check normality and randomness in distribution. The histogram of residuals and normal probability plot in Figure 5.2 shows that the residuals do not follow a normal distribution. The distribution has heavy tails and more positive residuals compared to the negative residuals. The deviation of the residuals from normality indicates the chances of predictions to go beyond the standard deviation measure given by the model.
5.1. Voltage prediction model

The scatter plot of the predicted values against the residuals (residual plot) is shown in figure 5.3. The points do not show an exact random distribution around mean. Also, the variance in predictions increases with the voltage. From the plot, it is observed that, at lower values, most residuals are positive, which indicates the tendency of the model to underestimate the minimum voltage required.

Comparing the results

To compare the performance of the GP model with the KNN regression model, the plot of predicted vs actual values of voltage in both the models are compared as shown in figure 5.4. Both the plots follow a linear trend. Table 5.2 shows the RMSE and explained variance score of the models on the test dataset. The explained variance score shows that the KNN regression model was able to capture 93.4% of the variance in the data while the GP model captures 98.4% variation in the data. The high value of explained variance score indicates a high correlation between the predicted output and the actual values. Generalized RMSE score of KNN model is 0.05, whereas the GP with Matérn $\nu=3/2$ kernel was able to give predictions with RMSE score of 0.025. The Gaussian Process Regression for voltage prediction was able to outperform the benchmark implementation using KNN.
5.2. Time prediction model

A dataset with 19274 data points was available for training the time prediction model, and it was divided into train, test and validation sets.

K-Nearest neighbours

The training and validation error was plotted against the number of neighbors to choose the optimal value of K. The elbow plot for time prediction model is shown in figure 5.5. Training error increases with an increase in the value of K. The validation error has its lowest value at K = 5; hence five was chosen as the optimal number of neighbors to train the final model. The model was able to make predictions with an RMSE value of 1.25 and an explained variance score of 0.81.

---

Figure 5.4: Plot of predicted voltage vs actual voltage

Table 5.2: RMSE and explained variance score of voltage prediction model on test dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>RMSE</th>
<th>Explained Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNN at K = 5</td>
<td>0.05</td>
<td>0.934</td>
</tr>
<tr>
<td>The Gaussian Process</td>
<td>0.025</td>
<td>0.984</td>
</tr>
</tbody>
</table>
The Gaussian processes regression

Gaussian processes suffer from cubic time complexity $O(N^3)$ where $N$ is the number of training samples. So, three computationally efficient sparse approximation algorithms are used to train the time prediction model, and they are:

1. Active learning to select the data points that exhibit the highest predictive variance
2. The informative vector machine (IVM)
3. Variational learning of inducing variables

These are greedy algorithms for sparse approximation. For active learning and the IVM algorithms, the number of data points needed for sparse representation has to be specified in advance. For choosing an optimal subset size, random subsets of different sizes were sampled, and a GP model was trained on each of these subsets.
5.2. Time prediction model

Figure 5.6 presents the root mean square error of the GP model for predicting time as a function of the subset size of data. From the plot, the error decreases as the subset size increases, and as the number of training points increases beyond 5000, the changes in error rates are trivial. Due to memory limitations, the effect of growing subset size beyond 7000 could not be tested. Considering the trade-off between the time complexity and prediction accuracy of the model, and anticipating the possibility of better results with subset size larger than 7000, optimal subset size was chosen to be 6000.

Active learning - selecting data points with maximum predictive variance

In this approximation, a subset of data points that exhibit maximum predictive variance is selected to train the Gaussian process regression model. The samples were chosen iteratively, and on each iteration, the data point that exhibits maximum predictive variance was added to the active set until the active set size was 6000.

Gaussian processes using different kernels was trained using the actively selected subset of data. Table 5.3 presents the values of root means squared error and explained variance score of the predictions on validation dataset. With different covariance functions, the variation in results are minor. The additive kernel of SE and Matérn\(_{\nu=3/2}\) is giving the highest explained variance score of 0.879. The RMSE has the lowest value of 1.03 for both Matérn\(_{\nu=3/2}\) kernel and the additive kernel SE and Matérn\(_{\nu=3/2}\). Because Matérn\(_{\nu=3/2}\) is simple compared to the additive covariance function, and the variation in scores is trivial, this kernel can be selected for the final model.

<table>
<thead>
<tr>
<th>Kernel</th>
<th>RMSE</th>
<th>Explained Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>SE</td>
<td>1.09</td>
<td>0.864</td>
</tr>
<tr>
<td>Matérn(_{\nu=3/2})</td>
<td>1.03</td>
<td>0.873</td>
</tr>
<tr>
<td>SE + Linear</td>
<td>1.05</td>
<td>0.867</td>
</tr>
<tr>
<td>SE * Linear</td>
<td>1.14</td>
<td>0.852</td>
</tr>
<tr>
<td>Matérn(_{\nu=3/2}) + Linear</td>
<td>1.04</td>
<td>0.872</td>
</tr>
<tr>
<td>Matérn(_{\nu=3/2}) * Linear</td>
<td>1.06</td>
<td>0.871</td>
</tr>
<tr>
<td>SE + Matérn(_{\nu=3/2})</td>
<td>1.03</td>
<td>0.879</td>
</tr>
</tbody>
</table>

The Informative Vector Machine

The sparse dataset for training the Gaussian process regression model was selected iteratively using the IVM algorithm. In each iteration, the data points which gives the maximum change in entropy on the addition to active set was selected. An active set of size 6000 selected using this algorithm was used in training the GP regression. Table 5.4 shows the prediction results of the model on validation dataset using different kernels. Matérn\(_{\nu=3/2}\) kernel and the composite kernel of Matérn\(_{\nu=3/2}\) and linear are providing the best results. Matérn\(_{\nu=3/2}\) kernel was selected to train the final model as it is simpler compared to the composite kernel. This model was able to make better predictions compared to the benchmark implementation using KNN as well as the sparse regression model trained using the active learning strategy.
5.2. Time prediction model

Table 5.4: Comparison of error statistic using different kernels - IVM.

<table>
<thead>
<tr>
<th>Kernel</th>
<th>RMSE</th>
<th>Explained Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>SE</td>
<td>1.108</td>
<td>0.858</td>
</tr>
<tr>
<td>Matérn$_{\nu=3/2}$</td>
<td>0.992</td>
<td>0.886</td>
</tr>
<tr>
<td>SE + Linear</td>
<td>1.108</td>
<td>0.858</td>
</tr>
<tr>
<td>SE * Linear</td>
<td>1.123</td>
<td>0.854</td>
</tr>
<tr>
<td>Matérn$_{\nu=3/2}$ + Linear</td>
<td>.992</td>
<td>0.886</td>
</tr>
<tr>
<td>Matérn$_{\nu=3/2}$ * Linear</td>
<td>1.03</td>
<td>0.877</td>
</tr>
<tr>
<td>SE + Matérn$_{\nu=3/2}$</td>
<td>0.987</td>
<td>0.887</td>
</tr>
</tbody>
</table>

Variational learning of inducing variables

In this algorithm, the inducing inputs used in training the GP regression model is consid-
ered as variational parameters. The inducing inputs and kernel hyperparameters are selected
jointly using continuous optimization by maximizing the lower bound of true log marginal
likelihood [39]. Python package Gpy [48] was used in implementing this approximation. Ta-
ble 5.5 presents the RMSE and explained variance score using different kernels and composite
kernels. The Sparse GP trained using Matérn$_{\nu=3/2}$ kernel is providing the best results, and
hence it was chosen as the kernel to train the final model.

Table 5.5: Comparison of error statistic using different kernels - variational learning.

<table>
<thead>
<tr>
<th>Kernel</th>
<th>RMSE</th>
<th>Explained Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>SE</td>
<td>1.372</td>
<td>0.783</td>
</tr>
<tr>
<td>Matérn$_{\nu=3/2}$</td>
<td>1.316</td>
<td>0.801</td>
</tr>
<tr>
<td>SE + Linear</td>
<td>1.384</td>
<td>0.779</td>
</tr>
<tr>
<td>SE * Linear</td>
<td>1.390</td>
<td>0.765</td>
</tr>
<tr>
<td>Matérn$_{\nu=3/2}$ + Linear</td>
<td>1.319</td>
<td>0.800</td>
</tr>
<tr>
<td>Matérn$_{\nu=3/2}$ * Linear</td>
<td>1.396</td>
<td>0.775</td>
</tr>
<tr>
<td>Matérn$_{\nu=3/2}$ + SE</td>
<td>1.317</td>
<td>0.801</td>
</tr>
</tbody>
</table>

The residuals from all the three sparse approximation are evaluated and compared to
analyze the performance of the models. Figure 5.7 shows the residual histogram and normal
probability plot for the models trained using the three sparse approximation algorithms.
From figure 5.7(a, b, c, d), the residual distribution of the model trained using both IVM and
active learning models highly deviate from the normal distribution and shows a non-linear
pattern. They have long tails, and this is an indication of high variance in the prediction. Fig-
ure 5.7(e, f) shows the histogram of residuals and the normal probability plot for the sparse
approximation using variational learning. The normal probability plot shows reasonable
linearity in the center, but both the tails show deviation from the linear trend, indicating a
normal distribution with tails.
5.2. Time prediction model

Figure 5.7: Residual distributions of time prediction model using different sparse approximation algorithms

Figure 5.8 shows the residual plots for sparse approximation models. All the plots show distribution around the horizontal axis but the variance in predictions is not constant.
5.2. Time prediction model

(a) Active learning

(b) IVM

(c) Variational learning

Figure 5.8: Residuals vs predicted values of log(time/s) for the time prediction model using sparse GP approximations

Comparing the results

To compare the results of the GP models with KNN, the plots of predicted vs actual values were plotted as shown in figure 5.9. The plot of KNN, IVM and active learning have points following a linear trend with more points closer to the diagonal line. An interesting pattern observed in all the plots is the points deviating from the linear trend in the region (-12,-10). This can be explained by the lack of an adequate amount of training data in this region. The model trained using variational approximation gives wrong predictions even for smaller values of time, whereas other models were able to make better predictions in this region. Also, in variational learning, the predicted values are concentrated between -25 and -17.5, and this could be because the algorithm is selecting more inducing points from this region and is failing to capture the entire variation.
5.2. Time prediction model

(a) KNN at $K = 5$

(b) Active learning

(c) IVM

(d) Variational Learning

Figure 5.9: Predicted vs actual plots of time prediction models

Table 5.6: RMSE and explained variance score of the models for time prediction

<table>
<thead>
<tr>
<th>Method</th>
<th>RMSE</th>
<th>Explained Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNN</td>
<td>1.25</td>
<td>0.81</td>
</tr>
<tr>
<td>Active learning</td>
<td>1.04</td>
<td>0.861</td>
</tr>
<tr>
<td>IVM</td>
<td>0.998</td>
<td>0.874</td>
</tr>
<tr>
<td>Variational learning</td>
<td>1.28</td>
<td>0.791</td>
</tr>
</tbody>
</table>

Finally, the models were evaluated using root mean squared error and explained variance score on test dataset. Table 5.6 presents the scores of all the regression models trained for predicting time. The KNN regression model for predicting time has an explained variance score of 0.81 and RMSE value of 1.25. From different sparse approximation algorithms, the IVM was able to produce the best results with RMSE of 0.998 and explained variance score of 0.874, which is a considerable improvement from the KNN regression model.
This chapter elaborates on the results from the analysis, the reliability of the results and the chosen methods. The discussions are focused on answering the following questions.

1. What conclusions can be drawn from the results presented in chapter 5?
2. Are the methods used in the research suitable for the given dataset?
3. How reliable are the prediction results?
4. What are the shortcomings of the work done?
5. How can the results be improved?

6.1 Results

The failure probability of a bitcell is determined to analyze the performance of a bitcell before using it in the memory design. In this thesis, two different models were proposed based on the performance matrices used in measuring the failure, and they are:

1. Voltage prediction model
2. Time prediction model

Both the models were trained using KNN and the Gaussian process regression algorithms. Chapter 5 presents the results obtained from the trained models.

Voltage prediction model

The minimum voltage needed in maintaining the stability of a bitcell is predicted using machine learning models. The KNN regression model was able to make predictions with an RMSE value of 0.05 and an explained variance score of 0.934. Table 5.1 shows the RMSE and explained variance score for predictions made using the GP regression model. This model outperforms the benchmark implementation with all covariance functions. Considering the error score, Matérn$ _{\nu=3/2} $ is the best candidate for the kernel. Whereas, with the explained
6.1. Results

The variance score, the additive kernel of Matérn$_{v=3/2}$ and linear provides the highest score. Instead of choosing a kernel solely based on one evaluation metrics, the combined scores, as well as the simplicity of the kernel, was considered leading to the choice of Matérn$_{v=3/2}$ for training the final model. The performance of the model was evaluated by making predictions on the test dataset. As shown in table 5.2, the model was able to make predictions with an explained variance score of 0.98 and an RMSE score of 0.025 which indicated the reduction in error rates by 50% compared to KNN.

The histogram of residuals in figure 5.2 shows that the residuals deviate from the normal distribution and are skewed with more positive values compared to the negatives. A residual is calculated as

$$\text{Residual} = \text{Observed value} - \text{Predicted value}$$

so, more positive residuals means that the predictions in most cases are lower than the actual value which shows the tendency of the model to underestimate the ‘minimum voltage’.

Figure 5.4 shows the plots of actual values of voltage against the predictions using both the trained models. The predicted values form both the models follows a linear trend. However, in the KNN regression model, the variance of the predictions increases with increase in the value of voltage. Especially when the voltage is ‘1.2’, most of the predictions are underestimated. The Gaussian process regression model was able to fit the data well compared to KNN and was able to make predictions with considerably less bias and reduced overall variance.

The failure of a bitcell can lead to failure of an entire memory. Hence high accuracy is expected in the predictions to use it in estimation. From the results, the GP regression model was able to make predictions with low RMSE value. Also, the bias on individual predictions is small, which makes them good to be used in estimating the performance of a bitcell.

Time prediction model

As discussed in section 1.1, one way to estimate the failure of a bitcell is to see if the read or write operation can be completed within a threshold time. The model, in this case, predicts time taken for read or write which in turn can be used in performance estimation.

K-nearest neighbors algorithm was used as benchmark implementation, and it was able to predict the time quite well with an RMSE value of 1.25 and an explained variance score of 0.81. The time prediction model was also trained using Gaussian process regression. Due to the high number of data points, three sparse approximation algorithms were implemented. Tables 5.3, 5.4 and 5.5 show the results of the three regression models with sparse approximations for the GP. From the evaluation scores, the first two models are outperforming the benchmark implementation while the KNN model was giving better scores compared to the sparse model trained using the variational learning approximation.

Based on the simplicity of the model and contemplating on both RMSE and explained variance score, Matérn$_{v=3/2}$ was chosen as the covariance function to train the final sparse approximation model in all the three cases. These models were evaluated on the test dataset, and their performance was compared to the benchmark implementation. Table 5.6 shows the generalized scores for the trained models. From the scores, it is observed that the IVM algorithm provides the predictions with the least value of root mean square error and the highest explained variance score and they are considerably better than the other models. The RMSE value is 0.99 for IVM compared to 1.04 for active learning, 1.28 in variational
learning and 1.25 for KNN. This means that the sparse GP model trained using the subset of data selected using IVM is able to fit the data for time prediction better than the other models.

From figure 5.9, time predictions from all the models are going wrong for the upper boundary values of the time. The prediction of log-transformed values of time beyond -12.5 is highly deviating from the linear trend. None of the models is able to capture the input pattern that leads to these values of the time. One possible reason for this pattern in prediction is the inadequate representation of training samples in those regions. Training the models with more data samples at higher values of time is one possible solution to get better predictions.

As discussed in section 1.1, the failure of a bitcell can lead to the failure of the memory. Hence the accuracy is a crucial consideration for the usability of the predictions. The IVM was able to give predictions with bias which is trivial enough so the prediction can be considered acceptable. But the variance is not constant, with some observed patterns at higher values of the time. The standard deviation measure provided by the Gaussian process gives an understanding of the associated uncertainty, and that can help in deciding the quality of prediction to be used in bitcell failure/ performance estimation. However, with the deviation of residuals from normality, the chances of predictions going wrong beyond standard deviation estimates cannot be discarded.

6.2 Method

To my knowledge, no machine learning implementations are available for the performance estimation of a bitcell. After analyzing the data, non-parametric models were selected as underlying data distribution was unknown. Unavailability of a benchmark implementation led to the choice of K-nearest neighbour algorithm considering its simplicity and the robustness. The choice of Gaussian processes to fit the data was highly influenced by the fact that GPs provide a full probabilistic model of the data where the associated uncertainty is also estimated along with the predictions.

In Gaussian process regression models, an important aspect is the selection of covariance function that is best suited for the available data. In this work, not much research has been put forth in the selection of the kernels. The most commonly used kernels and their combinations were tested on the available data, and the function providing the best scores was selected as the prior covariance function. To improve the quality and validity of predictions, more efficient methodology can be adopted in the kernel selection process and the implication of applying each of these kernels to the available dataset can be studied more extensively.

Three different sparse approximation algorithms were used to train the time prediction model, and the results were compared to select the best approximation. The main task in training the sparse GP models was the selection of an optimal subset of data. As shown in figure 5.6, a random subset selection procedure was adopted to select the optimal subset size. One drawback faced here was the memory constraint due to which the effect of the addition of samples beyond 7000 could not be tested. Also, in the subset selection process, the IVM and active learning algorithms adopted an iterative approach to add data to the training set. Due to memory limitations during the calculation of the kernel matrix, each iteration used a random subset of training data from which a datapoint which maximizes the selection criterion was added to the active set. Since the subset selection is random, the algorithm can select the best data points, but the chances of some data points that can provide valid information gain to the model getting not selected cannot be discarded.
6.3 Ethical considerations

This thesis was done in collaboration with Xenergic AB, and the company holds the intellectual property rights on this work.
In this thesis, the possibility of applying machine learning algorithms in performance estimation of a bitcell was investigated. Two different models were designed based on performance criteria measured for estimating a bitcell failure. The first model predicts the minimum voltage required to maintain the stability of the bitcell, and the second model was designed to predict the time taken for bitcell 'read' and 'write' operations. The experimental results for the above models were evaluated using the KNN and Gaussian process regression algorithms where KNN was considered as a benchmark. The voltage prediction model trained using the GPR provided good predictions with high explained variance score. Three sparse approximations of the GP, which includes the active learning strategy, the IVM and variational learning of inducing variables were implemented for the time prediction model. The results of the analysis indicate the IVM as the best approximation algorithm in terms of both root mean square error and explained variance score.

The predictions from the GP models along with the uncertainty measure, provide a reasonable estimate of the bitcell performance. But with the variance in predictions, the models are not reliable enough to replace existing statistical methods. But this research was able to conclude that machine learning models can be adapted in performance estimation of a bitcell. The predictions provided by the models can help in reducing the number of simulations needed by the in-house rare failure event estimator by a substantial factor. Also, these models can help in providing a quick overview of the behaviour of different input combinations in bitcell configuration during the design phase of an SRAM memory.


[24] Jiawei Han, Jian Pei, and Micheline Kamber. Data mining: concepts and techniques. Elsevier, 2011.


