Title
Hamming Distance Computation in Unreliable Resistive Memory

Author
Chen, Zehui

Publication Date
2018

Peer reviewed|Thesis/dissertation

University of California

Los Angeles

Hamming Distance Computation in Unreliable Resistive Memory

A thesis submitted in partial satisfaction

of the requirements for the degree

Master of Science in Electrical and Computer Engineering

by

Zehui Chen

2018

ABSTRACT OF THE THESIS

Hamming Distance Computation in Unreliable Resistive Memory

by

Zehui Chen

Master of Science in Electrical and Computer Engineering

University of California, Los Angeles, 2018

Professor Lara Dolecek, Chair

Enabled by new storage mediums, *Computation-in-Memory* is a novel architecture that has shown great potential in reducing the burden of massive data processing by bypassing the communication and memory access bottleneck. Suggested by Cassuto and Crammer, allowing for ultra-fast Hamming distance computations to be performed in resistive memory with low-level conductance measurements has the potential to drastically speed up many modern machine learning algorithms. Meanwhile, Hamming distance Computation-in-Memory remains a challenging task as a result of the non-negligible device variability in practical resistive memory. In this thesis, as a follow-up to the work from Cassuto and Crammer, we study memristor variability due to two distinct sources: resistance variation, and the non-deterministic write process. First, we introduce a technique for estimating the Hamming distance under resistance variation alone. Then, we propose error-detection and error-correction schemes to deal with non-ideal write process. We then combine these results to concurrently address both sources of memristor variabilities. In order to preserve the low latency property of *Computation-in-Memory*, all of our approaches rely on only a single vector-level conductance measurement. We use so-called inversion coding as a key

ingredient in our solutions and we prove the optimality of this code given the restrictions on bit-accessible information. Lastly, we demonstrate the efficacy of our approaches on the k-nearest neighbors classifier.

The thesis of Zehui Chen is approved.

Puneet Gupta

Jonathan Kao

Lara Dolecek, Committee Chair

University of California, Los Angeles

2018

Table of Contents

Acknowledgments

Chapter 1, Chapter 2, Chapter 3, Chapter 5, Section 4.1, Section 4.4, Section 4.5 and Appendix A are a version of [Zehui Chen, Clayton Schoeny, Lara Dolecek, "Hamming distance computation in unreliable resistive memory," manuscript submitted to TCOM].

Section 4.2 and Section 4.3 are a version of [Zehui Chen, Clayton Schoeny, Yuval Cassuto, Lara Dolecek, "A coding scheme for reliable in-memory Hamming distance computation," in Proceeding of Asilomar Conference of Signals, System, and Computers, Pacific Grove, CA, Oct.-Nov. 2017].

I would like to express my sincere gratitude to my advisor Professor Lara Dolecek for her support of my research. Her guidance helped me in all the time of research and writing of this thesis. Without her lead on research, I could not have touched this research topic that is both practically meaningful and interesting to myself.

I would like to thank the rest of my thesis committee: Professor Puneet Gupta and Professor Jonathan Kao, for their time and invaluable feedback.

My sincerely thanks goes to Professor Yuval Cassuto for his innovating prior work that opens up this research direction and for the insightful discussion on this topic.

I would like to thank Clayton Schoeny for his patiently help in the writing of all my work and for his enlightening ideas on my research questions.

The Research is supported in part by a grant from UC MEXUS and an NSF-BSF grant no.1718389.

CHAPTER 1

Introduction

With many emerging data-intensive applications, it has become imperative to have the means to store and quickly process vast amounts of high dimensional data. However, current computer architectures largely suffer from communication and memory access bottlenecks. Additionally, CMOS technology faces limited scalability issues [CZ14, HXN+15]. Resistive Random-Access Memory (ReRAM), also simply known as resistive memory, has been shown to have promising scalability with novel crossbar structure, and is thus a promising candidate for next generation none-volatile memories [SSS+08]. Enabled by this new technology, Computation-in-Memory (CIM) Architecture has been proposed, in which certain computations are performed in the physical memory itself [HXN+15]. This idea allows us to bypass the communication and memory access bottlenecks and can be used to speed up data-intensive applications.

Computing similarity metrics between vectors is a critical component in machine learning algorithms; image recognition and natural language processing are just some of many examples. It has already been shown that hashing higher dimensional data into binary space, and using Hamming distance as the distance metric is well suited for large-scale applications [KD09, NPF12]. Allowing for Hamming distance to be computed under the CIM Architecture is thus a promising technique to speed up many modern machine learning applications. Recently, a technique has been proposed to compute Hamming distance in resistive memory, assuming an ideal model for the memristors [CC15]. Following from their

work, in this thesis, we use more sophisticated models to improve and refine their scheme (to which we refer as Hamming distance Computation-in-Memory (HD-CIM)). The feasibility of HD-CIM is studied under two main (and complementary) sources of memristor variability: resistance variation, and the non-deterministic switching mechanism during the memristor write process. In order to preserve the low-latency property of HD-CIM, we assume *limited accessibility to information*. First, only vector-level conductance measurement can be used, and second, only one measurement can be used per Hamming distance computation. Dealing with the two sources of memristor variability is thus more challenging due to these limited accessibility assumptions, in particular, traditional ECCs based on bit-level information do not apply. Simple yet effective solutions are proposed to deal with these sources of memristor variability when the information that can be read is limited. As suggested in [CC15], the optimal code that maps a message to a constant weight codeword while preserving the Hamming distance is used as the foundation for our solutions. The efficacy of our approaches under these sources of memristor variability are studied in detail for the k-nearest neighbors classifier, one promising application for HD-CIM.

Below, we provide a brief outline of this thesis.

Chapter 2 provides background on resistive memory, memristors, and the two sources of memristor variability investigated in this thesis. The two sources of memristor variability, variability due to resistance variation and variability due to non-deterministic switching mechanism, are elaborated in detail. We highlight the trade-offs between these sources of variability and performance metrics of resistive memory in order to emphasize the necessity of concerning these sources of variability. The mathematical models of the adverse effects these variability sources are established in this chapter. The solutions that deal with the adverse effects the variability sources are at first studied separately.

Chapter 3 provides a solution to estimate the Hamming distance from a single conductance measurement in resistive memory where the resistances of memristors are formulated as Gaussian random variables. To generalize our solution, inversion coding is used to provide

a priori knowledge about vector weights. The estimation error probability and its bounds are also studied in Chapter 3 for the case with inversion coding and the case without inversion coding. In order to evaluate the effect of resistance variation on the end application, we introduce the average-case study framework. Using this framework, we present the performance of our estimation scheme on the k-nearest neighbors classifier as our main application.

Chapter 4 provides error-detection and error-correction schemes as solutions to bit-errors caused by the non-ideal write process. Single error detection is first studied and then we generalize the same idea to multiple errors detection. Two different error correction schemes are explored in this chapter. The exact error correction scheme is provided based on the inherent error localization capability of inversion coding and an extra code that correct erasures. Motivated by the error tolerant nature of our application, we also provides an approximate error correction scheme that have low latency and low overhead. The performance of this approximate error correction scheme is evaluated under the average-case study framework for the k-nearest neighbors classifier.

Chapter 5 combines results of Chapter 3 and Chapter 4 and provides a feasible scheme to estimate Hamming distance in resistive memory under the adverse effects of the two sources of memristor variability simultaneously. The joint effect of the two variability sources is first studied and the maximum *a posteriori* probability estimator is formulated. We then present a solution to the MAP estimator. Adapting the concept of approximate error correction scheme discussed in the last chapter, we provide a scheme that achieve estimation and error-detection/correction simultaneously.

Chapter 6 concludes this thesis. Appendix A provides necessary justification for one key approximation made in Chapter 3.

CHAPTER 2

Backgrounds

2.1 Memristors and Resistive Memory

In this thesis, we focus on resistive memory which uses a crossbar structure [VRK+09]. In crossbar resistive memory, a resistive switching device (also referred to as a memristor) is placed at the intersection of each row and column [VRK+09]. The logical states of "0" and "1" are represented by the internal resistance state of memristor, "High Resistance State (HRS)" and "Low Resistance State (LRS)," respectively. The state of the memristor can be programmed by applying different voltages to its terminals and can be sensed by measuring the corresponding current. In this work, under the same model used in [CC15], we are interested in inferring the Hamming distance between vectors from the conductance measurement between rows of memristors where the two vectors are stored. Figure 2.1 shows an example of the circuit representation that stores two vectors, $\boldsymbol{x} = (0, 0, 0, 1, 1)$ and $\boldsymbol{y} = (0, 1, 1, 0, 1)$, and the example conductance measurement between the two vectors, $G(\boldsymbol{x}, \boldsymbol{y})$. Using conductance in our calculations (instead of resistance) allows for a simple summation of each branch. We assume throughout the thesis that measurements between the two rows of memristors in resistive memory can be reliably performed.

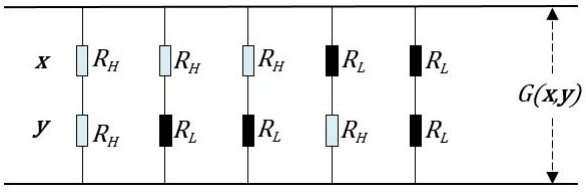


Figure 2.1: Example of an equivalent circuit for a measurement between two vectors

2.2 Variability Due to Resistance Variation

While an on/off ratio from ~ 10 to above 1000 has been shown in many ReRAM papers as proof of a large memory operation window, the variations of HRS and LRS are both common and significant [CL11]. Previous work on HD-CIM assumed constant valued LRS and HRS resistances [CC15]. It is therefore of interest to take resistance variation into account and develop corresponding schemes in order to perform Hamming distance computation in practical resistive memory. In practice, the resistance variation is affected by many operational parameters. It is reported that LRS variability depends on two main parameters, the write current limit I_{limit}, and the write pulse width [CL11]. A smaller I_{limit} favors low-power operation but increases the variation of LRS. A shorter pulse width favors fast speed operation but it also increases the variation of LRS. As a result, studying HD-CIM solutions that tolerate resistance variation not only makes HD-CIM feasible in practice but also provides useful insight into the trade-off between performance and operation parameters, i.e., pulse width and I_{limit}.

Many papers have shown that the resistance distribution of LRS and HRS are Gaussian-like [BKL+05, CLG+11, WLW+10]. The reported resistances of LRS and HRS are both positive and large, i.e., on the order $k\Omega$ and $M\Omega$. It is therefore reasonable to assume that the conductance also follows a Gaussian distribution if the corresponding resistance is Gaussian-like. We assume that the process variability in each memristor is identical and independent. For a memristor i, let L_i and H_i be random variables denoting the state "0" conductance and the state "1" conductance, respectively. We assume L_i and H_i follow the following Gaussian distributions:

$$L_i \sim \mathcal{N}(\mu_L, \sigma_L^2),\ H_i \sim \mathcal{N}(\mu_H, \sigma_H^2). \tag{2.1}$$

In the above model, μ_L and μ_H are the mean of state "0" and state "1" conductance, and σ_L^2 and σ_H^2 are the variance of state "0" and state "1" conductance, respectively. We define

$\epsilon = \mu_L/\mu_H$ which is also the "On/Off" resistance ratio, a key characteristic of memristor devices [VRK+09].

2.3 Variability Due to Non-deterministic Switching Mechanism

As is the case with resistance variation, the switching mechanism of the memristor is also non-deterministic. It is reported that the switching time of some memristors, e.g., TiO_2 cells, follows a log-normal distribution with the median switching time exponentially dependent on the external voltage [MRPC+11]. Either increasing the programming voltage or increasing the switching time will increase the switching probability. Meanwhile, increasing the programming voltage will increase energy consumption and increasing the switching time will slow down the write progress and increase energy consumption [NXX12, YPQ+11]. With finite switching time and programming volatage, instances of unsuccessful memristor switching are inevitable; studying the effect of the non-deterministic switching mechanism in HD-CIM and the corresponding solution is thus necessary to make HD-CIM practical.

Unsuccessful write operations lead to bit errors when the former state of a memristor is different from the data to be written. We thus model this variability as a binary symmetric channel (BSC) when writing to resistive memory, i.e., a write BSC. The vectors to be stored are viewed as the input to this channel, while the vectors actually written to the resistive memory are the output.

CHAPTER 3

Hamming Distance Estimation-In-Memory Under Resistance Variation

In 3.1, we provide a scheme (Hamming Distance Estimation-In-Memory) to estimate the Hamming distance between two vectors whose Hamming weights are *a priori* known, assuming the conductance of each memristor state is modeled as a random variable. In this chapter, we also provide a simplified scheme in the regime of small "On/Off" resistance ratios. In 3.2, an inversion coding technique is used to generalize Hamming distance Estimation-In-Memory to be applicable to arbitrary vectors with unknown Hamming weights, using a single conductance measurement. We then prove the optimality of this code and present a modified estimation scheme. In 3.3, we provide analysis of the estimation error and show that inversion coding also provides benefits in terms of estimation accuracy. In 3.4, the average-case study framework is introduced to study the effect of resistance variation on the k-nearest neighbors classifier when our estimation scheme is used.

3.1 Hamming Distance Estimation-In-Memory (HD-EIM)

For two vectors $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^n$ stored in resistive memory where conductances of the two states of each memristor are modeled as Gaussian random variables, the conductance between the two rows of memristors is as follows:

$$G(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{n}\Bigg[x_i y_i \frac{H_{i,\boldsymbol{x}}H_{i,\boldsymbol{y}}}{H_{i,\boldsymbol{x}}+H_{i,\boldsymbol{y}}} + x_i(1-y_i)\frac{H_{i,\boldsymbol{x}}L_{i,\boldsymbol{y}}}{H_{i,\boldsymbol{x}}+L_{i,\boldsymbol{y}}} + (1-x_i)y_i\frac{L_{i,\boldsymbol{x}}H_{i,\boldsymbol{y}}}{L_{i,\boldsymbol{x}}+H_{i,\boldsymbol{y}}} + (1-x_i)(1-y_i)\frac{L_{i,\boldsymbol{x}}L_{i,\boldsymbol{y}}}{L_{i,\boldsymbol{x}}+L_{i,\boldsymbol{y}}}\Bigg]. \tag{3.1}$$

Here, x_i, y_i denote the i-th entry of vectors $\boldsymbol{x}, \boldsymbol{y}$ respectively. $H_{i,x}$ and $H_{i,y}$ are random variables denoting the "1" state conductances of memristors that store x_i and y_i respectively, and they are modeled as Gaussian random variable in (2.1). $L_{i,x}$ and $L_{i,y}$ are similarly defined for the "0" state conductances. Equation (3.1) follows by summing up the conductances of each branch i, which is composed of the serial conductance of memristors that store x_i and y_i. Note that Equation (3.1) is analogous to Equation (1) in [CC15] with the substitution of our new variability model.

The following approximation can be made when $\mu_H >> \sigma_H$ and $\mu_L >> \sigma_L$:

$$\begin{aligned}
&\frac{H_{i,x}H_{i,y}}{H_{i,x}+H_{i,y}} \approx F_i \sim \mathcal{N}(\mu_H/2, \sigma_H^2/8), \quad \frac{L_{i,x}L_{i,y}}{L_{i,x}+L_{i,y}} \approx T_i \sim \mathcal{N}(\mu_L/2, \sigma_L^2/8),\\
&\frac{H_{i,x}L_{i,y}}{H_{i,x}+L_{i,y}} \approx S_i \sim \mathcal{N}\left(\frac{\mu_L}{1+\epsilon}, \frac{\sigma_L^2}{(1+\epsilon)^4}\right).
\end{aligned} \tag{3.2}$$

These approximations are made in order to facilitate further calculation (see Appendix A for justification of approximations). We define $\epsilon = \mu_L/\mu_H$ and provide examples of ϵ in Appendix A, Table A.1.

For two vectors $\boldsymbol{x}$, $\boldsymbol{y} \in \{0,1\}^n$, we define N_{00} to be the number of element pairs that have $x_i = y_i = 0$ for $i \in \{1, ..., n\}$. Similarly we define N_{01} to be the number of element pairs that have $x_i = 0, y_i = 1$, N_{10} to be the number of element pairs that have $x_i = 1, y_i = 0$, and N_{11} to be the number of element pairs that have $x_i = y_i = 1$. Using our approximations, the conductance measurement can be expressed as follows,

$$G(\boldsymbol{x},\boldsymbol{y}) \approx \bar{G}(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{N_{11}} F_i + \sum_{i=1}^{N_{01}+N_{10}} S_i + \sum_{i=1}^{N_{00}} T_i,$$

We normalize F_i, S_i, and T_i by $\mu_H/2$ and denote the normalized random variables by f_i, s_i, and t_i, (where $\epsilon = \mu_L/\mu_H$) yielding:

$$f_i \sim \mathcal{N}(1, \sigma_H^2/2\mu_H^2),\ s_i \sim \mathcal{N}\left(\frac{2\epsilon}{1+\epsilon}, \frac{4\sigma_L^2}{\mu_H^2(1+\epsilon)^4}\right),\ t_i \sim \mathcal{N}(\epsilon, \sigma_L^2/2\mu_H^2).$$

Similarly, we compute the normalized conductance measurement, $\tilde{G}(\boldsymbol{x}, \boldsymbol{y})$, as follows:

$$\tilde{G}(\boldsymbol{x}, \boldsymbol{y}) = \frac{\bar{G}(\boldsymbol{x}, \boldsymbol{y}) \times 2}{\mu_H} = \sum_{i=1}^{N_{11}} f_i + \sum_{i=1}^{N_{01}+N_{10}} s_i + \sum_{i=1}^{N_{00}} t_i.$$

Using elementary properties of independent Gaussian distributions, we have

$$\tilde{G}(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{N}\left(N_{11} + (N_{01}+N_{10})\frac{2\epsilon}{1+\epsilon} + N_{00}\epsilon, \frac{N_{11}\sigma_H^2}{2\mu_H^2} + \frac{(N_{01}+N_{10})8\sigma_L^2}{2\mu_H^2(1+\epsilon)^4} + \frac{N_{00}\sigma_L^2}{2\mu_H^2}\right). \quad (3.3)$$

We define w_x and w_y to be the *a priori* known Hamming weights of $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively. We define $D(\boldsymbol{x}, \boldsymbol{y})$ to be the Hamming distance between $\boldsymbol{x}$ and $\boldsymbol{y}$. Using the following facts,

$$N_{11} = [w_x + w_y - D(\boldsymbol{x}, \boldsymbol{y})]/2,\ N_{01} + N_{10} = D(\boldsymbol{x}, \boldsymbol{y}),\ N_{00} = n - N_{11} - N_{01} - N_{10}, \quad (3.4)$$

we can compute the following:

$$\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1+\epsilon}{(1-\epsilon)^2}[(1-\epsilon)(w_x + w_y) + 2n\epsilon - 2\tilde{G}(\boldsymbol{x}, \boldsymbol{y})]. \quad (3.5)$$

Note that Equation (3.5) is a substitution of the noisy $D(\boldsymbol{x}, \boldsymbol{y})$ into Equation (5) from [CC15]. Based on (3.3), (3.4) and (3.5), $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ has the following distribution:

$$\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{N}\left(D(\boldsymbol{x}, \boldsymbol{y}), \frac{2(1+\epsilon)^2 N_{11}\sigma_H^2}{\mu_H^2(1-\epsilon)^4} + \frac{(N_{01}+N_{10})16\sigma_L^2}{\mu_H^2(1+\epsilon)^2(1-\epsilon)^4} + \frac{2(1+\epsilon)^2 N_{00}\sigma_L^2}{\mu_H^2(1-\epsilon)^4}\right). \quad (3.6)$$

$\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ is an intermediate random variable computed using the conductance measurement from which we can estimate $D(\boldsymbol{x}, \boldsymbol{y})$. Note that $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ and $\tilde{G}(\boldsymbol{x}, \boldsymbol{y})$ will be rede-

fined per chapter for the appropriate context. We can now view the problem of estimating Hamming distance from a conductance measurement as a classic communication problem. The transmitter sends $D(\boldsymbol{x}, \boldsymbol{y})$ and the channel is Gaussian with zero mean and variance $\sigma^2(\tilde{D}(\boldsymbol{x}, \boldsymbol{y}))$, as specified in (3.6). The receiver sees $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ and estimates $D(\boldsymbol{x}, \boldsymbol{y})$ from the observation.

We note that $D(\boldsymbol{x}, \boldsymbol{y})$ takes only integer values from 0 to n. We also observe that for any two nearby distributions which center at D_H and $D_H + 1$, their variances only differ by a small amount when σ_H and σ_L are relatively close. Based on these observations, we propose an estimator $\hat{D}(\boldsymbol{x}, \boldsymbol{y}) = \text{nint}(\tilde{D}(\boldsymbol{x}, \boldsymbol{y}))$. We refer to this estimator as the *nearest integer estimator*. The nearest integer estimator completes the last step of HD-EIM, which estimates $D(\boldsymbol{x}, \boldsymbol{y})$ from a single measurement $G(\boldsymbol{x}, \boldsymbol{y})$.

From (3.6), we observe that $\sigma^2(\tilde{D}(\boldsymbol{x}, \boldsymbol{y}))$ does not have a clear relationship to the message $D(\boldsymbol{x}, \boldsymbol{y})$ which makes the analysis of the estimation error intractable. The approximation in (3.2), which leads to (3.6), is suitable for a variety of "On/Off" resistance ratios. However, smaller "On/Off" resistance ratios are usually reported in literature, e.g., $\epsilon = 0.01$ [CL11]. Therefore, we seek to further simplify our equations in order provide a single parameter characterization for the estimation error. When ϵ is small, the simplified approximations readily follow:

$$\begin{aligned} &\frac{H_{i,x}H_{i,y}}{H_{i,x}+H_{i,y}} \approx F_i \sim \mathcal{N}(\mu_H/2, \sigma_H^2/8), \quad \frac{L_{i,x}L_{i,y}}{L_{i,x}+L_{i,y}} \approx T_i \sim \mathcal{N}(\mu_L/2, \sigma_L^2/8), \\ &\frac{H_{i,x}L_{i,y}}{H_{i,x}+L_{i,y}} \approx S_i \sim \mathcal{N}(\mu_L, \sigma_L^2). \end{aligned} \tag{3.7}$$

With the simplified approximations, the distribution of $\tilde{G}(\boldsymbol{x}, \boldsymbol{y})$ and the calculation of $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ change accordingly:

$$\tilde{G}(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{N}\left(N_{11} + (N_{01}+N_{10})2\epsilon + N_{00}\epsilon, \frac{N_{11}\sigma_H^2}{2\mu_H^2} + \frac{(N_{01}+N_{10})8\sigma_L^2}{2\mu_H^2} + \frac{N_{00}\sigma_L^2}{2\mu_H^2}\right). \tag{3.8}$$

We then compute $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ as:

$$\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{1-3\epsilon}[(1-\epsilon)(w_x + w_y) + 2n\epsilon - 2\tilde{G}(\boldsymbol{x}, \boldsymbol{y})], \tag{3.9}$$

where

$$\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{N}\Big(D(\boldsymbol{x}, \boldsymbol{y}), \frac{2N_{11}\sigma_H^2 + 2N_{00}\sigma_L^2 + (N_{01}+N_{10})16\sigma_L^2}{\mu_H^2(1-3\epsilon)^2}\Big). \tag{3.10}$$

The estimation of $D(\boldsymbol{x}, \boldsymbol{y})$ is performed using the nearest integer estimator. This simplification is used in the remaining parts of this chapter, Chapter 4, and Chapter 5.

3.2 Inversion Coding

In the previous discussion, we proposed the HD-EIM scheme which estimates the Hamming distance between two vectors from a single conductance measurement, with the assumption that the weights of both vectors are *a priori* known. However, this assumption may not be valid for many applications. Therefore, in order to generalize our HD-EIM scheme to applications where vectors with arbitrary weights are of interest, we require a method to gain knowledge of vector weights. In this section we discuss two approaches to get vector weights before HD-EIM is performed: weight estimation and inversion coding. We briefly describe the idea of weight estimation and elaborate on the inversion coding technique.

In [CC15], where the ideal two-state conductance model of a memristor is considered, the weight of a vector can be computed from the measurement between memristor that store the vector itself and some preset vector, e.g., the all-1 vector. We can use this idea, in conjunction with the techniques described in 3.1, to estimate the weight of a vector in the presence of variability due to resistance variation. After the weights of two vectors are estimated, the Hamming distance can be then estimated using the HD-EIM scheme. This approach of estimating the weight of both vectors first, and then estimating the Hamming distance requires a total of three conductance measurements in order to complete one Hamming distance estimation. The extra two measurements introduce extra latency which is not

favored by frequent read applications. The overall estimation accuracy then also suffers from additional estimation errors when estimating the weights of vectors.

Alternatively, in applications where latency is the primary concern, we use the following coding technique to force every vector to have the same Hamming weight prior to the data being written to resistive memory, thus enabling Hamming distance Estimation-In-Memory with only one conductance measurement.

Auxiliary Code 1. *(cf. [CC15]). We define an inversion encoding of the vector* $\boldsymbol{x}$ *to be* $\boldsymbol{x}^{(c)} = [\boldsymbol{x}|\neg\boldsymbol{x}]$*, where* $\neg\boldsymbol{x}$ *is the bitwise complement of* $\boldsymbol{x}$ *and* $|$ *denotes concatenation.*

We refer to Auxiliary Code 1 as inversion coding throughout this thesis. Let us define the weight of a vector $\boldsymbol{x}$ as $w(\boldsymbol{x})$. With inversion coding, we have $w(\boldsymbol{x}^{(c)}) = n, \forall \boldsymbol{x} \in \{0,1\}^n$. With inversion coded vectors stored in resistive memory, the weights are known to be n, thus HD-EIM can be readily used. By the nature of inversion coding, we have $D(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)}) = 2D(\boldsymbol{x}, \boldsymbol{y})$. This relationship can thus be used to estimate $D(\boldsymbol{x}, \boldsymbol{y})$ from the $\tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)})$ using the following equation of the redefined $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$:

$$\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2}\tilde{D}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)}), \tag{3.11}$$

where

$$\tilde{D}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)}) = \frac{1}{1-3\epsilon}[2n(1-\epsilon) + 4n\epsilon - 2\tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)})]. \tag{3.12}$$

The random variables $\tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)})$ denote the normalized conductance measurements between two rows that store the coded vectors $\boldsymbol{x}^{(c)}$ and $\boldsymbol{y}^{(c)}$; $\tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)})$ will be redefined per chapter and section for the appropriate context. Adapting (3.8), we have the distribution of $\tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)})$ as follows:

$$\tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)}) \sim \mathcal{N}\left(N'_{11} + (N'_{01} + N'_{10})2\epsilon + N'_{00}\epsilon, \frac{N'_{11}\sigma_H^2}{2\mu_H^2} + \frac{(N'_{01} + N'_{10})8\sigma_L^2}{2\mu_H^2} + \frac{N'_{00}\sigma_L^2}{2\mu_H^2}\right). \tag{3.13}$$

Here N'_{gh} is defined to be the number of coordinates having bit g in $\boldsymbol{x}^{(c)}$ and bit h in $\boldsymbol{y}^{(c)}$,

for $g, h \in \{0, 1\}$. We therefore have:

$$\tilde{D}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)}) \sim \mathcal{N}\left(D(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)}), \frac{2N'_{11}\sigma_H^2 + 2N'_{00}\sigma_L^2 + (N'_{01} + N'_{10})16\sigma_L^2}{\mu_H^2(1-3\epsilon)^2}\right), \tag{3.14}$$

and

$$\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{N}\left(D(\boldsymbol{x}, \boldsymbol{y}), \frac{2N'_{11}\sigma_H^2 + 2N'_{00}\sigma_L^2 + (N'_{01} + N'_{10})16\sigma_L^2}{4\mu_H^2(1-3\epsilon)^2}\right). \tag{3.15}$$

We can thus estimate $D(\boldsymbol{x}, \boldsymbol{y})$ from $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ using the nearest integer estimator $\hat{D}(\boldsymbol{x}, \boldsymbol{y}) = \text{nint}(\tilde{D}(\boldsymbol{x}, \boldsymbol{y}))$, thus adapting the HD-CIM scheme to the case where an inversion coding is used. We now show the optimality of inversion coding in terms of its redundancy among all constant weight codes.

Lemma 1. *Define a code to be a injective mapping that maps a message $\boldsymbol{x} \in \{0, 1\}^a$ to a codeword $\bar{\boldsymbol{x}} \in \{0, 1\}^b$. Let $D(\boldsymbol{x}, \boldsymbol{y})$ denote the Hamming distance between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. We define a code to be Hamming distance preserving if and only if $D(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}) = f\left(D(\boldsymbol{x}, \boldsymbol{y})\right)$ for some bijective function f. We also define a constant weight code to be a code that satisfies the property $w(\bar{\boldsymbol{x}}) = w_0, \forall \bar{\boldsymbol{x}} \in \{0, 1\}^b$ and some constant w_0. The necessary conditions for a constant weight code to be Hamming distance preserving are:*

$$w_0 \geq a, \ b \geq 2a.$$

Proof. Define $\mathcal{D}_1$ to be the range of $D(\boldsymbol{x}, \boldsymbol{y})$ for $\boldsymbol{x}, \boldsymbol{y} \in \{0, 1\}^a$. We have $|\mathcal{D}_1| = a + 1$. Define $\mathcal{C}_1$ to be the set of length-b binary vectors that have weight w_0. Also define $\mathcal{C}_2$ to be the set of length-b codewords generated by any constant weight code with weight w_0. Due to the nature of codes, $\mathcal{C}_2 \subseteq \mathcal{C}_1$. Define $\mathcal{D}_2$ to be the range of $D(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ for $\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}} \in \mathcal{C}_1$, define $\mathcal{D}_3$ to be the range of $D(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ for $\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}} \in \mathcal{C}_2$. The following relationship can be easily verified, $|\mathcal{D}_2| = \min(w_0 + 1, b - w_0 + 1)$. In order for the code to be Hamming distance preserving, i.e., f to be a bijective function, we need $|\mathcal{D}_3| = |\mathcal{D}_1| = a + 1$. Since $\mathcal{C}_2 \subseteq \mathcal{C}_1$, we have $|\mathcal{D}_3| = a + 1 \leq |\mathcal{D}_2| = \min(w_0 + 1, b - w_0 + 1)$, which proves our necessary conditions. ■

Due to the limited read accessibility to resistive memory, there is no direct access to $\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}$. For any constant weight code we could use, we can only estimate the Hamming distance between between pairs of codewords using HD-EIM scheme. It is therefore necessary for our constant weight code to be Hamming distance preserving so that we can directly recover the Hamming distance between the corresponding pairs of messages. Since the rate of this code is a/b, Lemma 1 implies that the maximum rate of a code of this type is $1/2$. The inversion coding indeed has rate $1/2$ and is thus optimal in term of redundancy.

3.3 Estimation Error Probability and Bounds

In previous sections, we have introduced two methods of conductance variability to the HD-CIM method of [CC15]: one in which vectors with known weights are stored and the other in which inversion coded vectors are stored. Due to resistance variation, estimation errors can occur when determining the Hamming distance between vectors. In this section, the estimation error probability is studied for the two cases and the results are compared.

Lemma 2. *When two vectors $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^n$ with known weight are stored in resistive memory, relying on approximation (3.7), we can estimate $D(\boldsymbol{x}, \boldsymbol{y})$ using $\hat{D}(\boldsymbol{x}, \boldsymbol{y}) = nint(\tilde{D}(\boldsymbol{x}, \boldsymbol{y}))$, with $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ computed from (3.9), and the normalized conductance measurement $\tilde{G}(\boldsymbol{x}, \boldsymbol{y})$ from (3.8). Then, the conditional estimation error probability has the following bound:*

$$P(\hat{D}(\boldsymbol{x}, \boldsymbol{y}) \neq D(\boldsymbol{x}, \boldsymbol{y}) \mid D(\boldsymbol{x}, \boldsymbol{y})) \leq 2Q\left(\frac{1}{2\sqrt{\beta(n + 7D(\boldsymbol{x}, \boldsymbol{y}))}}\right), \tag{3.16}$$

where $\beta = \frac{2\max(\sigma_L^2, \sigma_H^2)}{\mu_H^2(1-3\epsilon)^2}$ and $Q(\cdot)$ is the Q-function, i.e., $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty \exp(-\frac{u^2}{2}) du$.

Proof. From $\hat{D}(\boldsymbol{x}, \boldsymbol{y}) = nint(\tilde{D}(\boldsymbol{x}, \boldsymbol{y}))$ and (3.10), the probability of erroneous estimation

can be calculated as:

$$P(\hat{D}(\boldsymbol{x},\boldsymbol{y}) \neq D(\boldsymbol{x},\boldsymbol{y}) \mid D(\boldsymbol{x},\boldsymbol{y})) = 2Q\left(\frac{1}{2\sigma(\tilde{D}(\boldsymbol{x},\boldsymbol{y}))}\right),$$

Define $\sigma^2_{max} = \max(\sigma^2_H, \sigma^2_L)$. The standard deviation of $\tilde{D}(\boldsymbol{x},\boldsymbol{y})$ can be upper bounded:

$$\begin{aligned}\sigma(\tilde{D}(\boldsymbol{x},\boldsymbol{y})) &= \sqrt{\frac{2N_{11}\sigma^2_H + 2N_{00}\sigma^2_L + (N_{01}+N_{10})16\sigma^2_L}{\mu^2_H(1-3\epsilon)^2}} \\ &\leq \sqrt{\frac{2N_{11}\sigma^2_{max} + 2N_{00}\sigma^2_{max} + (N_{01}+N_{10})16\sigma^2_{max}}{\mu^2_H(1-3\epsilon)^2}} = \sqrt{\beta(n+7D(\boldsymbol{x},\boldsymbol{y}))}.\end{aligned}$$

Using elementary properties of the Q-function, Lemma 2 is proved. ■

Lemma 2 provides us with a single parameter relation between β and the estimation error probability. The parameter β serves as a measure of device reliability and examples of β are provided in Appendix A Table A.1. It is also observed that the estimation error probability is largely dependent on the Hamming distance. A smaller Hamming distance is associated with a smaller estimation error probability. This property is well suited for classification problems in which objects with smaller distances are of interest.

We also provide a bound of the unconditional estimation error probability of HD-EIM when vectors with known weights are stored.

Corollary 1. *The unconditional estimation error probability of HD-EIM, when vectors with known weights are stored, is upper bounded as:*

$$P(\hat{D}(\boldsymbol{x},\boldsymbol{y}) \neq D(\boldsymbol{x},\boldsymbol{y})) \leq 2Q\left(\frac{1}{4\sqrt{2\beta n}}\right).$$

Proof. Follows immediately from $D(\boldsymbol{x},\boldsymbol{y}) \leq n, \forall\, \boldsymbol{x},\boldsymbol{y}$ and Lemma 2. ■

This simple bound gives insight into the trade-off between β, the device reliability and n, the vector length (scalability). For instance, we can achieve the same level of Hamming distance calculation accuracy with larger resistance variation by shortening the vectors.

We next adapt Lemma 2 and Corollary 1 to the case where inversion coding is used.

Lemma 3. *When two inversion coded vectors, $\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)} \in \{0,1\}^{2n}$, corresponding to two original vectors $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^n$, are stored in resistive memory, relying on approximation (3.7), we can estimate $D(\boldsymbol{x}, \boldsymbol{y})$ using $\hat{D}(\boldsymbol{x}, \boldsymbol{y}) = nint(\tilde{D}(\boldsymbol{x}, \boldsymbol{y}))$, with $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ computed from (3.11), and the normalized conductance measurement $\tilde{G}(\boldsymbol{x}, \boldsymbol{y})$ from (3.13). Then, the conditional estimation error probability is upper bounded as:*

$$P(\hat{D}(\boldsymbol{x}, \boldsymbol{y}) \neq D(\boldsymbol{x}, \boldsymbol{y}) \mid D(\boldsymbol{x}, \boldsymbol{y})) \leq 2Q\left(\frac{1}{\sqrt{2\beta(n + 7D(\boldsymbol{x}, \boldsymbol{y}))}}\right), \tag{3.17}$$

where $\beta = \frac{2\max(\sigma_L^2, \sigma_H^2)}{\mu_H^2(1-3\epsilon)^2}$.

Proof. The proof of this lemma is similar to proof of Lemma 2 with $\sigma^2(\tilde{D}(\boldsymbol{x}, \boldsymbol{y}))$ following from (3.15) in conjunction with the following properties of inversion coding:

$$N'_{11} = N'_{00} = N_{11} + N_{00}, \;\; N'_{01} = N'_{10} = N_{01} + N_{10}.$$

■

We also provide a simple bound of the estimation error probability of HD-EIM when inversion coded vectors are stored.

Corollary 2. *The unconditional estimation error probability of HD-EIM, when inversion coded vectors are stored, is upper bounded as:*

$$P(\hat{D}(\boldsymbol{x}, \boldsymbol{y}) \neq D(\boldsymbol{x}, \boldsymbol{y})) \leq 2Q\left(\frac{1}{4\sqrt{\beta n}}\right)$$

Proof. Follows immediately from $D(\boldsymbol{x}, \boldsymbol{y}) \leq n, \forall\, \boldsymbol{x}, \boldsymbol{y}$ and Lemma 3. ■

The bounds in Lemmas 2 and 3 are tight when $\sigma_H^2 = \sigma_L^2$. We observe that when these bounds are tight, the estimation error probability using coded vectors is smaller than the

one using uncoded vectors. This observation shows that inversion coding also improves the estimation accuracy, in addition to its ability to generalize HD-EIM to vectors with unknown weights. Note that additional coding, e.g., encoding $\boldsymbol{x}$ to be $[\boldsymbol{x}|\neg\boldsymbol{x}|\boldsymbol{x}|\neg\boldsymbol{x}]$, further improves estimation accuracy but may not be favorable in terms of redundancy. The single inversion coding technique is the maximum rate constant weight code that preserves Hamming distance, thus allowing HD-EIM to be efficiently performed with a single conductance measurement. However, additional coding only improves estimation accuracy with diminishing returns.

3.4 An Average-Case Study on the k-NN Classifier Under Computation Noise Using HD-EIM

In the previous chapter we analyzed the estimation error of the Hamming distance Computation-in-Memory scheme from [CC15] under device variability due to resistance variation. Due to resistance variation, the Hamming distance between vectors has to be estimated and could lead to estimation error. From the application point of view, the estimation error can be viewed as computation noise when Hamming distance computation is performed. We have provided a bound on the conditional error probability. Meanwhile, erroneous Hamming distance computations may not necessarily lead to erroneous results due to the error-tolerant nature of some applications, e.g., the k-nearest neighbors classifier. In this section, we focus on the k-nearest neighbors classifier as our main application and provide analysis on how this computation noise affects the classification accuracy. Throughout this whole section, we assume binary attributes (vectors) are inversion coded and stored in resistive memory. We next introduce the framework we use to analyze the k-nearest neighbors classifier under computation noise.

The average-case analysis framework, introduced by Pazzani and Sarrett [PS92], is a useful theoretical framework to understand the behavior of learning algorithms. This frame-

work is based on the formal computation of the expected accuracy of a learning algorithm for a certain fixed class of concepts [OY97]. In section 3.1, we use this framework to formally compute the expected accuracy of the k-NN classifier to learn the m-of-n/ℓ concept, a boolean threshold function, under computation noise. The m-of-n/ℓ concept is defined by the number of relevant attributes (n), the number of irrelevant attributes (ℓ), and the threshold (m). Under this concept, an instance is positive if m or more relevant attributes exist and is negative if fewer than m relevant attributes exist. Note that n is redefined in this section in order to be consistent with [OY97]. We assume that relevant and irrelevant attributes occur with probabilities p and q, respectively. The k-NN classifier classifies an instance as positive if more than half of its k nearest neighbors are positive. When a tie occurs, the classifier randomly decides the class. We compute the expected accuracy when the bound is tight, i.e., $\sigma_H^2 = \sigma_L^2$ and the computation noise is parameterized by β, as stated in Lemma 3. The expected classification accuracy is a function of $n, m, p, q, k, N,$ and β, where N is the size of the training space. We use the next example to illustrate the idea of the k-nearest neighbors classifier and how an erroneous Hamming distance computation may not lead to erroneous classification.

Example 1. *In this example, we are interested in the classification of the testing instance based on the 5 training instances using a k-nearest neighbors classifier. The 1 testing instance and 5 training instances are listed in the next Table. We set k to be 3 for the k-NN classifier. The training instances are generated according to the 4-of-8/0 concept and we use 1/0 to mark positive/negative class labels.*

Table 3.1: k-NN Example

Testing Instance					
Attribute	11111000				
Training Instances					
Instance #	1	2	3	4	5
Class Label	1	0	1	0	0
Attribute	11111100	11100000	11110100	11000000	11000100
Hamming Distance to Testing Case	1	2	2	3	4

In the error-free case, i.e., all Hamming distances are computed correctly, the k-NN classifier classifies the testing instance as positive because its 3 nearest neighbors (instance #1, #2, and #3) have class labels 1, 0, and 1 respectively. Positive is the correct class label for the testing instance based on the 4-of-8/0 concept.

Now we consider the case where Hamming distance computation is prone to estimation error due to resistance variation. We take a special case where the Hamming distance between the testing instance (11111000) and training instance #2 (11100000) is erroneously computed to be 3. Note that there are two training instances (#2 and #4) at distance 3 w.r.t. the testing instance; the k-NN classifier will randomly choose one to be the nearest neighbor of the testing instance. Observe that although the Hamming distance computation is erroneous, the classification result is still positive because the 2 closest training instance (#1 and #3) are both positive. This example illustrate how erroneous Hamming distance computation may not lead to erroneous classification.

3.4.1 Formal Calculation of Expected Accuracy

In [OY97], Okamoto and Nobuhiro used the average-case study framework to analyze noisy attributes and class labels. We adapt some of their equations to the scenario involving computation noise. For the calculations of expected accuracy that have already been done by Okamoto and Nobuhiro, we simply state their results here (we refer readers to [OY97] for further details). We modify the equations to incorporate the computation noise and to reflect the fact that in our model, the attributes and class labels are noise-free.

The probability that an instance consists of x relevant attributes and y irrelevant attributes can be calculated as [OY97]:

$$P_{oc}(x,y) = \binom{n}{x}\binom{l}{y}p^x(1-p)^{n-x}q^y(1-q)^{l-y}.$$

Then the expected classification accuracy can be calculated as [OY97]:

$$A(k) = \sum_{y=0}^{l} \left[\sum_{x=0}^{m-1} P_{oc}(x,y)(1 - P_{pos}(k,x,y)) + \sum_{x=m}^{n} P_{oc}(x,y) P_{pos}(k,x,y) \right],$$

where $P_{pos}(k,x,y)$ represents the probability that the k-NN classifier classifies an arbitrary test instance with x relevant attributes and y irrelevant attributes as positive.

To calculate $P_{pos}(k,x,y)$, we first calculate $P_{dp}(x,y,e)$ ($P_{dn}(x,y,e)$, resp.) which is the probability that an arbitrary positive (negative, resp.) training instance has Hamming distance e from the arbitrary testing instance $t(x,y) \in I(x,y)$. The Hamming distance e is assumed to be estimated by HD-EIM scheme from an observation $\tilde{e}$. $I(x,y)$ represents the set of instances in which x relevant attributes and y irrelevant attributes simultaneously occur. $P_{dp}(x,y,e)$ and $P_{dn}(x,y,e)$ can be represented as:

$$P_{dp}(x,y,e) = \sum_{\hat{e}=0}^{n} \sum_{\hat{y}=0}^{l} \sum_{\hat{x}=m}^{n} P_{oc}(\hat{x},\hat{y}) P_{dis}(x,y,\hat{x},\hat{y},\hat{e}) P_H(e,\hat{e}), \tag{3.18}$$

and

$$P_{dn}(x,y,e) = \sum_{\hat{e}=0}^{n} \sum_{\hat{y}=0}^{l} \sum_{\hat{x}=0}^{m-1} P_{oc}(\hat{x},\hat{y}) P_{dis}(x,y,\hat{x},\hat{y},\hat{e}) P_H(e,\hat{e}). \tag{3.19}$$

$P_{dis}(x,y,\hat{x},\hat{y},e)$ is the probability that an arbitrary instance in $I(\hat{x},\hat{y})$ has Hamming distance e from an arbitrary instance in $I(x,y)$. $P_H(e,\hat{e})$ is the probability that the original Hamming distance $\hat{e}$ is estimated to be e.

In order to compute $P_H(e,\hat{e})$, we consider the following scenario where the observation $\tilde{e}$ is a random variable with distribution $\mathcal{N}(\hat{e}, \frac{1}{2}\beta(n + 7\hat{e}))$. This corresponds to the case that two attribute vectors with Hamming distance $\hat{e}$ are inversion coded and then stored in resistive memory. For $e = \text{nint}(\tilde{e})$ if $0 \leq \tilde{e} \leq n$, $e = 0$ if $\tilde{e} < 0$, and $e = n$ if $\tilde{e} > n$, using elementary properties of Gaussian distribution, $P_H(e,\hat{e})$ is calculated as follows:

$$P_H(e,\hat{e}) = \begin{cases} 1 - Q\left(\frac{\frac{1}{2}-\hat{e}}{\sqrt{\frac{1}{2}\beta(n+7\hat{e})}}\right) & \text{if} \quad e = 0, \\ Q\left(\frac{n-\frac{1}{2}-\hat{e}}{\sqrt{\frac{1}{2}\beta(n+7\hat{e})}}\right) & \text{if} \quad e = n, \\ Q\left(\frac{e-\hat{e}-\frac{1}{2}}{\sqrt{\frac{1}{2}\beta(n+7\hat{e})}}\right) - Q\left(\frac{e-\hat{e}+\frac{1}{2}}{\sqrt{\frac{1}{2}\beta(n+7\hat{e})}}\right) & \text{otherwise.} \end{cases}$$

The remaining equations in this subsection are stated directly from [OY97]; we include them here for completeness. $P_{dis}(x, y, \hat{x}, \hat{y}, e)$ can be calculated as follows [OY97]:

$$P_{dis}(x, y, \hat{x}, \hat{y}, e) = \sum_{(z_r, z_i) \in S} \frac{\binom{x}{z_r}\binom{n-x}{\hat{x}-z_r}}{\binom{n}{\hat{x}}} \frac{\binom{y}{z_i}\binom{l-y}{\hat{y}-z_i}}{\binom{l}{\hat{y}}},$$

where S is a set of all pairs of z_r and z_i that satisfy the following conditions:

$$\max(0, x+\hat{x}-n) \leq z_r \leq \min(x, \hat{x}), \max(0, y+\hat{y}-l) \leq z_i \leq \min(y, \hat{y}), z_r+z_i = \frac{x+y+\hat{x}+\hat{y}-e}{2}.$$

$P_{pos}(k, x, y)$ can be then calculated.

$$P_{pos}(k, x, y) = \sum_{d=0}^{n+l} \sum_{a=0}^{k-1} \sum_{b=k-a}^{N-a} P_{num}(x, y, d, a, b) P_{sp}(x, y, d, a, b).$$

where

$$P_{num}(x, y, d, a, b) = \binom{N}{a}\binom{N-a}{b} P_l(x, y, d)^a \times P_d(x, y, d)^b (1 - P_l(x, y, d) - P_d(x, y, d)^{N-a-b},$$

$$P_l(x, y, d) = \sum_{e=0}^{d-1} (P_{dp}(x, y, e) + P_{dn}(x, y, e)),$$

$$P_d(x, y, d) = P_{dp}(x, y, d) + P_{dn}(x, y, d),$$

and

$$P_{sp}(k, x, y, d, a, b)$$
$$= \sum_{u=0}^{a} \sum_{v=0}^{b} \left\{ P_{lp}^{(u)}(a, u) P_{dp}^{(v)}(b, v) \sum_{w=\lceil \frac{k+1}{2} \rceil - u}^{v} \left\{ P_{dp}^{(w)}(k, a, b, v, w) + \frac{1}{2} P_{dp}^{(w)}(k, a, b, v, \frac{k}{2} - u) \right\} \right\},$$

where

$$P_{lp}^{(u)}(a,u) = \binom{a}{u}\left(\frac{\sum_{e=0}^{d-1} P_{dp}(x,y,e)}{P_l(x,y,d)}\right)^u \left(\frac{\sum_{e=0}^{d-1} P_{dn}(x,y,e)}{P_l(x,y,d)}\right)^{a-u},$$

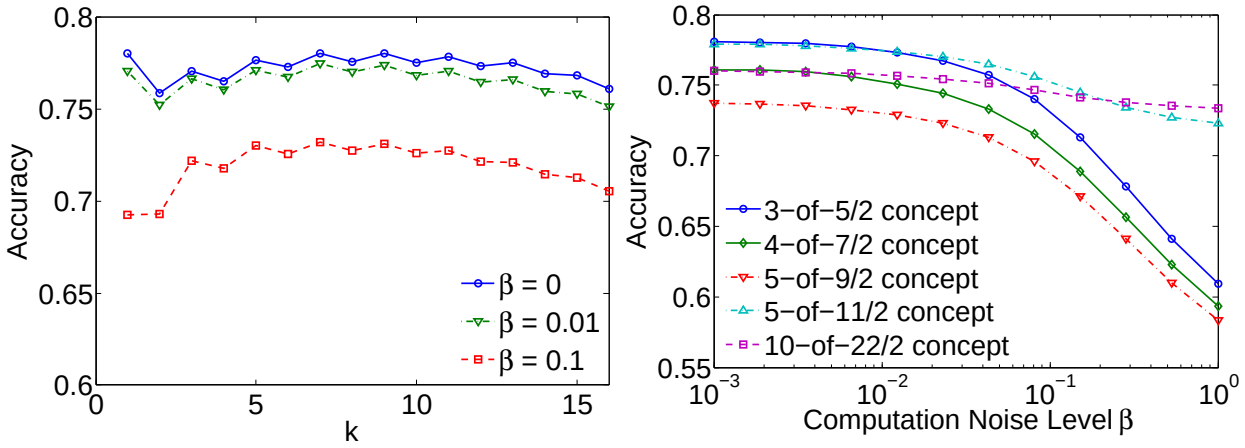
$$P_{dp}^{(v)}(b,v) = \binom{b}{v}\left(\frac{P_{dp}(x,y,d)}{P_d(x,y,d)}\right)^v \left(\frac{P_{dn}(x,y,d)}{P_d(x,y,d)}\right)^{b-v},$$

$$P_{dp}^{(w)}(k,a,b,v,w) = \frac{\binom{v}{w}\binom{b-v}{k-a-w}}{\binom{b}{k-a}}.$$

This ends our derivation for the expected accuracy.

3.4.2 Average-Case Study Results Using HD-EIM

First we study the effects of different levels of computation noise on the 3-of-5/2 concept. In Figure 3.1a, we fix N to be 32 and we report two device reliability (noise-level), $\beta = 0.01$ and $\beta = 0.1$, for k spanning from 1 to 16.



(a) 3-of-5/2 concept under computation noise

(b) computation noise v.s. max accuracy

Figure 3.1: Average-Case Study Results on Computation Noise

In Figure 3.1a, the upper line is the theoretical result for the noise-free classification accuracy, which is consistent with the results in [OY97]. The two lower lines are the theoretical accuracy with different device parameter β. When k is an even number, it is observed that the classification accuracy of the k-NN classifier drops remarkably due to the randomness

when a tie occurs. Figure 3.1a shows that the computation noise decreases the classification accuracy for all k and the amount of decrement largely depends on the noise level β.

We next show the effects of computation noise for a wide range of noise levels on a variety of different concepts in Figure 3.1b. The classification accuracy of k-NN at each noise level for each concept is chosen to be the maximum accuracy out of the range $2 \leq k \leq 16$.

In Figure 3.1b, β ranges from 10^{-3} to 1 in order to see a clear trend. However, from a realistic point of view, $\beta = \frac{2\max(\sigma_L^2, \sigma_H^2)}{\mu_H^2(1-3\epsilon)^2} = 1$ is unlikely to occur in a real device. We observe that for small β, i.e., $\beta < 10^{-2}$, the influence of the estimation error is negligible. It is interesting to note that the impact of the estimation error decreases as the dimension of concepts increases, which could be beneficial for applications with many attributes, e.g., $n = 22$.

CHAPTER 4

Error-Detection and Error-Correction Scheme Under Write BSC

In this chapter we consider memristor variability due to unreliable write operations. The adverse effect of unreliable write operation can be modeled as a write BSC and therefore can be viewed as attribute noise in learning problems. In [OY97], the authors provide an average-case study and observed that attribute noise decreases classification accuracy. In order to keep the low latency property of HD-CIM, we are restricted to the conductance measurements between rows of memristors. Therefore, traditional ECCs based on entry-wise access can not be used to deal with the write BSC and this fact motivates us to provide error-detection and error-correction schemes based on conductance measurements only. In this chapter, error-detection schemes are explored in 4.1 and 4.2. Two different error-correction schemes are explored in 4.3 and 4.4. In 4.5, we present the average-case study on k-nearest neighbor classifier using one of our proposed error-correction scheme. Throughout this chapter, we assume the memristors have no resistance variation, i.e., $\sigma_H^2 = \sigma_L^2 = 0$. We again assume inversion coded vectors are stored in the resistive memory.

4.1 Single Error Detection

Define $\boldsymbol{x}^{(c)}$ and $\boldsymbol{y}^{(c)}$ to be two inversion coded vectors corresponding to two length-n vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. Let $\tilde{\boldsymbol{x}}^{(c)}$ and $\tilde{\boldsymbol{y}}^{(c)}$ be the noisy vectors actually stored due to the write noise (BSC).

We first establish necessary equations to compute the Hamming distance between $\boldsymbol{x}$ and $\boldsymbol{y}$ from a measurement between rows of memristors that store $\tilde{\boldsymbol{x}}^{(c)}$ and $\tilde{\boldsymbol{y}}^{(c)}$. Define the normalized conductance measurement to be $\tilde{G}(\tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)})$. As resistance variability is not considered in this section, $\tilde{G}(\tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)})$ is a constant rather than a random variable. We thus have:

$$\tilde{G}(\tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)}) = \tilde{N}'_{11} + (\tilde{N}'_{10} + \tilde{N}'_{01})\frac{2\epsilon}{1+\epsilon} + \tilde{N}'_{00}\epsilon. \tag{4.1}$$

Here $\tilde{N}'_{gh}$ is defined to be the number of coordinates having bit g in $\tilde{\boldsymbol{x}}^{(c)}$ and bit h in $\tilde{\boldsymbol{y}}^{(c)}$, for $g, h \in \{0, 1\}$.

A variable denoting the normalized conductance measurement between rows of memristors that store $\boldsymbol{x}^{(c)}$ and $\boldsymbol{y}^{(c)}$ can be defined as:

$$\tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)}) = N'_{11} + (N'_{10} + N'_{01})\frac{2\epsilon}{1+\epsilon} + N'_{00}\epsilon. \tag{4.2}$$

Here N'_{gh} is defined to be the number of coordinates having bit g in $\boldsymbol{x}^{(c)}$ and bit h in $\boldsymbol{y}^{(c)}$, for $g, h \in \{0, 1\}$.

We again use an intermediate variable $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ to denote the result calculated from the measurement as follows:

$$\tilde{D}(\tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)}) = \frac{1+\epsilon}{(1-\epsilon)^2}[2n(1-\epsilon) + 4n\epsilon - 2\tilde{G}(\tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)})]. \tag{4.3}$$

$$\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2}\tilde{D}(\tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)}) = \frac{1+\epsilon}{(1-\epsilon)^2}[n + n\epsilon - \tilde{G}(\tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)})]. \tag{4.4}$$

Note that when no error occurs, $\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) = D(\boldsymbol{x}, \boldsymbol{y})$. Equation (4.3) is adapted from Equation (3.5) when inversion coding is used and write BSC is considered.

It is useful to denote the difference in conductance measurements between the noisy and the noise-free case as:

$$\Delta\tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)}, \tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)}) = \tilde{G}(\tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)}) - \tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)}).$$

Similarly, let us define $\Delta D(\boldsymbol{x}, \boldsymbol{y}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$, which is later used to characterize bit errors, as follows:

$$\Delta\tilde{D}(\boldsymbol{x}, \boldsymbol{y}, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}) = \tilde{D}(\boldsymbol{x}, \boldsymbol{y}) - D(\boldsymbol{x}, \boldsymbol{y}) = -\frac{(1+\epsilon)\Delta\tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)}, \tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)})}{(1-\epsilon)^2}. \tag{4.5}$$

We now calculate how $\Delta\tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)}, \tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)})$ and $\Delta\tilde{D}(\boldsymbol{x}, \boldsymbol{y}, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})$ are affected by a single bit error, i.e., $t = 1$. Due to the symmetry of this problem, there are only four different fundamental types of errors for a pair of elements $x_i^{(c)}$ and $y_i^{(c)}$, $(0,0) \to (0,1)$, $(0,1) \to (0,0)$, $(0,1) \to (1,1)$, and $(1,1) \to (0,1)$, which we denote as error types A, B, C, and D, respectively. All other error types are expressed in terms of these four error types. Each of these error types affects the relationship between $\{N'_{00}, N'_{01}, N'_{10}, N'_{11}\}$ and $\{\tilde{N}'_{00}, \tilde{N}'_{01}, \tilde{N}'_{10}, \tilde{N}'_{11}\}$. For example, an error that changes $(0,0)$ into $(0,1)$ will result in $\tilde{N}'_{01} = N'_{01} + 1$, $\tilde{N}'_{00} = N'_{00} - 1$, $\tilde{N}'_{11} = N'_{11}$ and $\tilde{N}'_{10} = N'_{10}$. Table 4.1 lists the resulting values of $\Delta\tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)}, \tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)})$ and $\Delta\tilde{D}(\boldsymbol{x}, \boldsymbol{y}, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})$ for each error type. The above two variables are abbreviated as $\Delta\tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)})$ and $\Delta\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ since the relative relationship is clear.

Table 4.1: The 4 Types of Errors

Error Type	$(x_i^{(c)}, y_i^{(c)}) \to (\tilde{x}_i^{(c)}, \tilde{y}_i^{(c)})$	$\Delta\tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)})$	$\Delta\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$
A	(0,0) → (0,1)	$\frac{2\epsilon}{1+\epsilon} - \epsilon$	$\frac{-\epsilon}{1-\epsilon}$
B	(0,1) → (0,0)	$-\left(\frac{2\epsilon}{1+\epsilon} - \epsilon\right)$	$\frac{\epsilon}{1-\epsilon}$
C	(0,1) → (1,1)	$-\left(\frac{2\epsilon}{1+\epsilon} - 1\right)$	$\frac{-\epsilon}{1-\epsilon} - 1$
D	(1,1) → (0,1)	$\frac{2\epsilon}{1+\epsilon} - 1$	$\frac{\epsilon}{1-\epsilon} + 1$

We use the following lemma to detect a single bit error.

Lemma 4. *If exactly one of $\tilde{\boldsymbol{x}}^{(c)}$ or $\tilde{\boldsymbol{y}}^{(c)}$ contains a single bit error, i.e., $D(\boldsymbol{x}^{(c)}, \tilde{\boldsymbol{x}}^{(c)}) + D(\boldsymbol{y}^{(c)}, \tilde{\boldsymbol{y}}^{(c)}) = 1$ and $0 < \epsilon < \frac{1}{2}$, then we can detect that $\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) \neq D(\boldsymbol{x}, \boldsymbol{y})$.*

Proof. If no error is present, we have $D(\boldsymbol{x}, \boldsymbol{y}) = \tilde{D}(\boldsymbol{x}, \boldsymbol{y})$, i.e., $\Delta\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) = 0$. For $0 < \epsilon < \frac{1}{2}$, each error type will cause the value of $\Delta\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ to be a non-integer and in turn lead to non-integer $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ (since $D(\boldsymbol{x}, \boldsymbol{y})$ is always an integer and $\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) = D(\boldsymbol{x}, \boldsymbol{y}) + \Delta\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$). A single bit error can thus be detected by first computing $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ and checking whether it is an integer or not. ■

This process of error detection is later referred to as an *integer check*. When error-free, $\hat{D}(\boldsymbol{x}, \boldsymbol{y}) = \tilde{D}(\boldsymbol{x}, \boldsymbol{y}) = D(\boldsymbol{x}, \boldsymbol{y})$ where $\hat{D}(\boldsymbol{x}, \boldsymbol{y})$ is the estimated result from the intermediate variable $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$.

Lemma 4 suggests that a single bit error is detectable under certain reasonable constraints on ϵ. However, the error type can not be uniquely determined from $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$. For example, a vector with $D(\boldsymbol{x}, \boldsymbol{y}) = D_H$ incurring error type B would result in an identical value for $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ as a vector with $D(\boldsymbol{x}, \boldsymbol{y}) = D_H - 1$ incurring error type D. Note that this multiple solution result is due to the nature of the underlying problem and is thus unavoidable if only vector-level information is used.

4.2 Multiple Errors Detection

So far we have shown that a single bit error in a pair of inversion coded vectors can be detected by checking whether $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ is an integer or not. Next, we generalize the idea of an *integer check* to multiple errors.

Lemma 5. *If together $\tilde{\boldsymbol{x}}^{(c)}$ and $\tilde{\boldsymbol{y}}^{(c)}$ contain an odd number of bit errors t, i.e., $D(\boldsymbol{x}^{(c)}, \tilde{\boldsymbol{x}}^{(c)}) + D(\boldsymbol{y}^{(c)}, \tilde{\boldsymbol{y}}^{(c)}) = t$ (odd), then we conclude that $\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) \neq D(\boldsymbol{x}, \boldsymbol{y})$ for the following constraint on ϵ:*

$$0 < \epsilon < \frac{1}{t+1}.$$

Proof. Define $\boldsymbol{e}_{i,p}, i \in \{1, ..., t\}, p \in \{\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)}\}$ to be the error vectors corresponding to the i-th error in the vector indexed by p. For clarity, the superscript in p is omitted. For example, if the i-th bit flip is at the j-th position of $\boldsymbol{x}^{(c)}$, $\boldsymbol{e}_{i,\boldsymbol{x}}$ is the all-0 vector with 1 at the j-th

position and $\boldsymbol{e}_{i,\boldsymbol{y}}$ is an all-0 vector. (We permit error at the j-th position in $\boldsymbol{y}_c$ but there will be another pair of error vectors corresponding to it.) We further define $\Delta\tilde{D}(\boldsymbol{e}_i)$ to be the change in output induced by $\boldsymbol{e}_{i,\boldsymbol{x}}$ and $\boldsymbol{e}_{i,\boldsymbol{y}}$, i.e.,

$$\Delta\tilde{D}(\boldsymbol{e}_i) = -\frac{(1+\epsilon)\Delta\tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)}, \boldsymbol{x}^{(c)} + \boldsymbol{e}_{i,\boldsymbol{x}}, \boldsymbol{y}^{(c)} + \boldsymbol{e}_{i,\boldsymbol{y}})}{(1-\epsilon)^2}.$$

The following relation is observed:

$$\begin{aligned}
&\Delta D(\boldsymbol{x}, \boldsymbol{y}, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}) \\
&= \sum_{i=1}^{t} \{-\frac{(1+\epsilon)[\tilde{G}(\boldsymbol{x}^{(c)} + \boldsymbol{e}_{i,\boldsymbol{x}}, \boldsymbol{y}^{(c)} + \boldsymbol{e}_{i,\boldsymbol{y}}) - \tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)})]}{(1-\epsilon)^2}\} \\
&= \sum_{i=1}^{t} \Delta D(\boldsymbol{e}_i).
\end{aligned}$$

Each $\Delta D(\boldsymbol{e}_i)$ can be viewed as the change of the output for a single bit error, thus assuming the same value as the last column in Table 4.1. For $0 < \epsilon < \dfrac{1}{t+1}$, $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ has a non-integer value because any odd t choices from those values of $D(\boldsymbol{e}_i)$ will be summed to $\Delta D(\boldsymbol{x}, \boldsymbol{y}, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})$, where $0 < |\Delta D(\boldsymbol{x}, \boldsymbol{y}, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})| \pmod 1 < 1$. Here and elsewhere, the operation $x \pmod 1$ is defined as the fractional part of x. As a result, any odd number of bit errors can be detected by an *integer check*.

■

We next present the result for an even number of errors, t.

Lemma 6. *We denote the fraction of errors that are detectable for even t as $r_d(t)$. If together $\tilde{\boldsymbol{x}}^{(c)}$ and $\tilde{\boldsymbol{y}}^{(c)}$ contain an even number of bit errors t, i.e., $D(\boldsymbol{x}^{(c)}, \tilde{\boldsymbol{x}}^{(c)}) + D(\boldsymbol{y}^{(c)}, \tilde{\boldsymbol{y}}^{(c)}) = t$ (even), and the channel parameter satisfies $0 < \epsilon < \dfrac{1}{t+1}$, then*

$$r_d(t) = 1 - \frac{\binom{t}{t/2}}{2^t}.$$

Proof. For an even number of errors, some error patterns are undetectable. We define $t_{A,C}$

to be the number of errors of either type A or C and similarly for $t_{B,D}$. Notice that error types A and C (and error types B and D) have the same integer parts in $\Delta D(\boldsymbol{e}_i)$, thus contributing equally to $|\Delta D(\boldsymbol{x}, \boldsymbol{y}, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})| \pmod 1$. The analysis on $t_{A,C}$ and $t_{B,D}$ thus covers all the possible error combinations. The error pattern is undetectable when $t_{A,C} = t_{B,D} = t/2$. By viewing this error detection problem as a fair-coin flipping problem in which $t/2$ heads and $t/2$ tails occur in t trials, we calculate the probability of an undetectable pattern occurring to be $\binom{t}{t/2}/2^t$. When $t_{A,C} \neq t_{B,D}$, let $t' = |t_{A,C} - t_{B,D}|$. We have $|\Delta D(\boldsymbol{x}, \boldsymbol{y}, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})| \pmod 1 = \frac{\epsilon t'}{1-\epsilon}, 0 < t' \leq t$. With $0 < \epsilon < \dfrac{1}{t+1}$, we have $0 < |\Delta D(\boldsymbol{x}, \boldsymbol{y}, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})| \pmod 1 < 1$ which directly allows for error detection. ■

Lemma 5 and Lemma 6 show that multiple errors are sometimes detectable by an *integer check*. Note that by an *integer check* alone, we are unable to differentiate the detected multiple errors and the detected single error.

4.3 Exact Error Correction with Additional Coding

From previous discussion, we have error detection schemes to detect erroneous computation of $D(\boldsymbol{x}, \boldsymbol{y})$ by an *integer check* of $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$. Our ultimate goal is to recover the correct Hamming distance between vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. In this section, we provide an error correction scheme based on acquiring the full error profile, i.e., error location and error value, in vectors $\tilde{\boldsymbol{x}}^{(c)}$ and $\tilde{\boldsymbol{y}}^{(c)}$ and we call this scheme the Exact Hamming scheme. Error localization is done by comparing with preset vectors while utilizing property of inversion coding. $\boldsymbol{x}^{(c)}$ and $\boldsymbol{y}^{(c)}$ can thus be successfully recovered relying on extra error correction code.

4.3.1 Error Localization

Claim 1. *Define L to be the set of vectors $\boldsymbol{l}_i \in \{0,1\}^{2n}, 1 \leq i \leq n$ whose all bits are one except the the i-th and the $(i+n)$-th bits. Also define $\mathbf{1}$ to be the all-1s vector with length $2n$. Assume error is detected in a pair of vectors $\tilde{\boldsymbol{x}}^{(c)}$ and $\tilde{\boldsymbol{y}}^{(c)}$, with $n+1$ pairwise measurements*

between rows of memristors that store $\tilde{\boldsymbol{x}}^{(c)}$ and vectors in $L \cup \mathbf{1}$, we can narrow the location of each error (if any) to two positions, i and $i+n$ except rare cases. Same applies to $\tilde{\boldsymbol{y}}^{(c)}$.

Without loss of generality, we assume there are bit errors in $\tilde{\boldsymbol{x}}^{(c)}$. With the measurement of $\tilde{G}(\tilde{\boldsymbol{x}}^{(c)}, \mathbf{1})$, we compute $\tilde{D}(\tilde{\boldsymbol{x}}, \mathbf{1})$ using Equation 4.4. We then perform n measurements between rows of memristor that store $\tilde{\boldsymbol{x}}^{(c)}$ and each of the vectors in L to compute $\tilde{D}(\tilde{\boldsymbol{x}}, \boldsymbol{l}_i)$ using the same equation. Error localization is achieved by computing $\Delta D_i(\tilde{\boldsymbol{x}}^{(c)}, \mathbf{1}, \tilde{\boldsymbol{x}}^{(c)}, \boldsymbol{l}_i) = \tilde{D}(\tilde{\boldsymbol{x}}, \boldsymbol{l}_i) - \tilde{D}(\tilde{\boldsymbol{x}}, \mathbf{1})$ for all $1 \leq i \leq n$. Let $I_{error} = \{i \in I | \Delta D_i(\tilde{\boldsymbol{x}}^{(c)}, \mathbf{1}, \tilde{\boldsymbol{x}}^{(c)}, \boldsymbol{l}_i) = \frac{2\epsilon}{1-\epsilon} + 2\} \cup \{i \in I | \Delta D_i(\tilde{\boldsymbol{x}}^{(c)}, \mathbf{1}, \tilde{\boldsymbol{x}}^{(c)}, \boldsymbol{l}_i) = \frac{2\epsilon}{1-\epsilon}\}$. There is an error either at position i or at position $i+n$ for each $i \in I_{error}$.

By Auxiliary Code 1, for a given $\boldsymbol{l}_i$, if both the i-th and the $(i+n)$-th positions of $\tilde{\boldsymbol{x}}^{(c)}$ are error-free, the error patterns are (1,1) $\rightarrow$ (0,1) and (0,1) $\rightarrow$ (0,0) which result in $\Delta D_i(\tilde{\boldsymbol{x}}^{(c)}, \mathbf{1}, \tilde{\boldsymbol{x}}^{(c)}, \boldsymbol{l}_i) = \frac{2\epsilon}{1-\epsilon} + 1$. If a bit error occurs at either the i-th or the $(i+n)$-th position of $\hat{\boldsymbol{x}}_c$, then the corresponding error patterns are two (1,1) $\rightarrow$ (0,0) which lead to $\Delta D_i(\tilde{\boldsymbol{x}}^{(c)}, \mathbf{1}, \tilde{\boldsymbol{x}}^{(c)}, \boldsymbol{l}_i) = \frac{2\epsilon}{1-\epsilon} + 2$ for bit flip from 1 to 0 or two (0,1) $\rightarrow$ (0,0) with $\Delta D_i(\tilde{\boldsymbol{x}}^{(c)}, \mathbf{1}, \tilde{\boldsymbol{x}}^{(c)}, \boldsymbol{l}_i) = \frac{2\epsilon}{1-\epsilon}$ for bit flip from 0 to 1. If errors occur at both the i-th or the $(i+n)$-th positions of $\hat{\boldsymbol{x}}_c$, we are unable to localize those two errors.

4.3.2 Bit Value Reading

Using the same idea of comparing with preset vectors, we are able to read bit-value at arbitrary location using conductance measurements between rows of memristor.

Claim 2. *Define B to be the set of vectors $\boldsymbol{b}_i \in \{0,1\}^{2n}, 1 \leq i \leq 2n$ whose bits are all one except the i-th bit. The i-th bit value of a given vector, say $\tilde{\boldsymbol{y}}^{(c)}$, can be inferred from two pairwise measurements between, $\tilde{\boldsymbol{y}}^{(c)}$ and $\boldsymbol{b}_i$; $\tilde{\boldsymbol{y}}^{(c)}$ and $\mathbf{1}$.*

Two measurements, $\tilde{G}(\tilde{\boldsymbol{y}}^{(c)}, \mathbf{1})$ and $\tilde{G}(\tilde{\boldsymbol{y}}^{(c)}, \boldsymbol{b}_i)$, are taken and the corresponding $\tilde{D}(\tilde{\boldsymbol{y}}, \mathbf{1})$ and $\tilde{D}(\tilde{\boldsymbol{y}}, \boldsymbol{b}_i)$ are computed using equation 4.4. The inference of the i-th bit value in $\hat{\boldsymbol{y}}_c$ is based on $\Delta D_i(\tilde{\boldsymbol{y}}^{(c)}, \mathbf{1}, \tilde{\boldsymbol{y}}^{(c)}, \boldsymbol{b}_i) = \tilde{D}(\tilde{\boldsymbol{y}}, \boldsymbol{b}_i) - \tilde{D}(\tilde{\boldsymbol{y}}, \mathbf{1})$. For $y_i^{(c)} = 1$, the error pattern is (1,1)

$\rightarrow$ (1,0) with $\Delta D_i(\tilde{\boldsymbol{y}}^{(c)}, \mathbf{1}, \tilde{\boldsymbol{y}}^{(c)}, \boldsymbol{b}_i) = \dfrac{\epsilon}{1-\epsilon} + 1$ and for $y_i^{(c)} = 0$, the error pattern is (0,1) $\rightarrow$ (0,0) with $\Delta D_i(\tilde{\boldsymbol{y}}^{(c)}, \mathbf{1}, \tilde{\boldsymbol{y}}^{(c)}, \boldsymbol{b}_i) = \dfrac{\epsilon}{1-\epsilon}$.

4.3.3 Error Correction with Additional Coding

For each error we localized using technique described in 4.3.1, we have ambiguity among two locations. One of these locations is in the first half of the erroneous vector and the other is in the second half. Based on the underlying inversion coding, the values at those two positions are complement of each other when error-free. We can thus focus on recovering the value of the first position by viewing it as an erasure. Here we propose adding extra redundancy by encoding the first half of $\boldsymbol{x}^{(c)}$ using a standard erasure correcting code. After error is detected, these parity bits can be accessed using technique described in 4.3.2. Because of this additional coding, we are able to correct every erasure that is conveyed by the error localization step. With the bit-value information of the corresponding vector (acquired using technique in 4.3.2), we are able to recover the Hamming distance between the uncorrupted vectors. The error correction capability of this Exact Hamming Scheme depends on the erasure correction capability of this exact code.

The caveat here is that for this erasure correcting code, the Hamming distance may not be preserved. As a result, error correction capability requires the ability to measure the conductance between sub-rows of memristor that store two sub-vectors in the resistive memory.

4.4 Soft Hamming Scheme for Single Error Correction

The Exact Hamming scheme can correctly find the exact error location and error value thus achieving the goal of recovering the original Hamming distance. However, it suffers from many aspects. First, the error correction capability of Exact Hamming scheme requires extra erasure correction code which adds redundancy and also requires sub-vector reading.

Second, the error localization procedure incurs many costly read operations between rows of memristors. Lastly, the error localization procedure may fail if rare error patterns occur and the erasure correction procedure may fail if there is bits error in the corresponding parity bit. These drawbacks of the Exact Hamming schemes make it not favorable in terms of its cost in space and time.

We notice that some of the potential application of HD-CIM, e.g., the k-nearest neighbor classifier, have certain error tolerant capability. This observation motivate us to provide an approximate error correction scheme focused on correcting the most frequent error, the single bit error. This approximate error correction scheme will improve the performance of our application under the write BSC while keep a small error-detection/error-correction latency. This scheme is based on the following corollary.

For single bit error that occurs, although the error type can not be uniquely determined by an *integer check*, with further constrains on ϵ, we are able to determine whether the error is of type {B,D} or {A,C} .

Corollary 3. *If* $D(\boldsymbol{x}^{(c)}, \tilde{\boldsymbol{x}}^{(c)}) + D(\boldsymbol{y}^{(c)}, \tilde{\boldsymbol{y}}^{(c)}) = 1$ *and* $0 < \epsilon < \frac{1}{3}$*, then we conclude that* $\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) \neq D(\boldsymbol{x}, \boldsymbol{y})$ *and we determine whether the error is of type {B,E} or {A,C}.*

Proof. The ϵ range here, $0 < \epsilon < \frac{1}{3}$, falls within the range of ϵ in Lemma 4, thus a single bit error is detectable. Since $| \frac{\epsilon}{1-\epsilon} | < \frac{1}{2}$, if the error belongs to type {B,D}, $\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) - \text{nint}(\tilde{D}(\boldsymbol{x}, \boldsymbol{y})) < 1/2$. Similarly, if the error belongs to type {A,C}, we have $\text{nint}(\tilde{D}(\boldsymbol{x}, \boldsymbol{y})) - \tilde{D}(\boldsymbol{x}, \boldsymbol{y}) < 1/2$. This relationship between $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ and its nearest integer uniquely determine whether the error is of type {A,C} or {B,D}. ■

We say $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ lies on the right-hand-side (or left-hand-side) of its nearest integer if $\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) - \text{nint}(\tilde{D}(\boldsymbol{x}, \boldsymbol{y})) < 1/2$ (or $\text{nint}(\tilde{D}(\boldsymbol{x}, \boldsymbol{y})) - \tilde{D}(\boldsymbol{x}, \boldsymbol{y}) < 1/2$).

As a result of Corollary 3, if $0 < \epsilon < \frac{1}{3}$, for the detected single bit error, we have two possible Hamming distances that could be the original one. We summarize the candidate

original Hamming distance as follows:

$$\begin{aligned}
&\text{If } \tilde{D}(\boldsymbol{x},\boldsymbol{y}) - \text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y})) < 1/2,\\
&\quad \text{then } D(\boldsymbol{x},\boldsymbol{y}) = \text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y})) \text{ or } D(\boldsymbol{x},\boldsymbol{y}) = \text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y})) - 1;\\
&\text{if } \text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y})) - \tilde{D}(\boldsymbol{x},\boldsymbol{y}) < 1/2,\\
&\quad \text{then } D(\boldsymbol{x},\boldsymbol{y}) = \text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y})) \text{ or } D(\boldsymbol{x},\boldsymbol{y}) = \text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y})) + 1.
\end{aligned}$$

We now seek to provide a correction scheme to the detected single bit error. We first realize that choosing randomly between the two candidate solutions is functionally the same as always choosing $\hat{D}(\boldsymbol{x},\boldsymbol{y}) = \text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y}))$. Therefore, choosing randomly would not be wise because it does not take advantage of the extra information we can get from Corollary 3.

We propose the following scheme for error-detection and error-correction.

$$\begin{aligned}
&\text{If} \quad \tilde{D}(\boldsymbol{x},\boldsymbol{y}) = \text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y})), \text{then } \hat{D}(\boldsymbol{x},\boldsymbol{y}) = \text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y}));\\
&\text{if} \quad \tilde{D}(\boldsymbol{x},\boldsymbol{y}) - \text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y})) < 1/2, \text{then } \hat{D}(\boldsymbol{x},\boldsymbol{y}) = \text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y})) - 1/2;\\
&\text{if} \quad \text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y})) - \tilde{D}(\boldsymbol{x},\boldsymbol{y}) < 1/2, \text{then } \hat{D}(\boldsymbol{x},\boldsymbol{y}) = \text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y})) + 1/2.
\end{aligned} \tag{4.6}$$

The previous scheme is derived by taking the average of the two candidate Hamming distances after an error is classified as either type {A,C} or {B,D}. As this error-detection and correction scheme has the possibility to give non-integer Hamming distance as the result, we refer it as the Soft Hamming scheme. Analysis of this Soft Hamming scheme on the k-nearest neighbors classifier will be provided in the next chapter.

4.5 An Average-Case Study on the k-NN Classifier Under Attribute Noise Using the Soft Hamming Scheme

In order to incorporate the Soft Hamming scheme into the average-case study framework, we assume all error patterns, except the undetectable errors, are detectable and can be

separated into the two classes—{A,C} or {B,D}—. We also assume ϵ is sufficiently small and the Soft Hamming scheme is used to correct the detected error.

The next example helps to illustrate how the Soft Hamming Scheme can be used to improve classification accuracy under the write BSC.

Example 2. *In this example we use the same setup as Example 1. Consider the case that under the adverse effect of write BSC, the attribute of training instance #3 (11110100) is changed to 01110100. Without the Soft Hamming scheme, the Hamming distance between training instance #3 and the testing instance is 3. Both instance #3 (class: positive) and instance #4 (class: negative) now have distance 3 w.r.t. the testing instance. Choosing randomly among instance #3 and #4 to be the third nearest neighbor of the testing instance, the k-NN classifier have 50% chance to have erroneous classification.*

If we implement the Soft Hamming scheme, the error pattern $(1,1) \to (0,1)$ *can be detected when computing the Hamming distance between the testing instance and training instance #3. The Soft Hamming scheme will therefore set the Hamming distance between them to be 2.5 based on previous discussions. As a result, the k-NN classifier can correctly classify the testing instance as positive.*

4.5.1 Formal Calculation of Expected Accuracy

We define an augmented variable $e^{(s)} = 2\hat{D}(\boldsymbol{x}, \boldsymbol{y})$, where $\hat{D}(\boldsymbol{x}, \boldsymbol{y})$ is the result of the Soft Hamming scheme. For example, $e^{(s)} = 3$ corresponds to the case that either $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ is on the left-hand-side of 1 or $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ is on the right-hand-side of 2. As another example, $e^{(s)} = 4$ corresponds to the case that $\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) = \hat{D}(\boldsymbol{x}, \boldsymbol{y}) = 2$. Most calculations from Section IV are applicable in this context by setting $\beta = 0$ since resistance variation is not considered. We recalculate $P_{dp}^{(s)}(x, y, e^{(s)})$ ($P_{dn}^{(s)}(x, y, e^{(s)})$, resp.) which is the probability of an arbitrary positive (negative, resp.) training instance having distance $e^{(s)}$ from the arbitrary testing instance $t(x, y) \in I(x, y)$. This calculation is separated into two cases. We assume, in this subsection, an arbitrary attribute is flipped with probability p.

In the first case, we consider an even value for $e^{(s)}$ where no error is detected. From previous discussion, no error is detected when the number of bit flips from 0 to 1, s, is equals to the number of bit flips from 1 to 0, r. When $e^{(s)}$is even, $P_{dp}^{(s)}(x, y, e^{(s)})$ and $P_{dn}^{(s)}(x, y, e^{(s)})$ can be expressed as:

$$P_{dp}^{(s)}(x, y, e^{(s)}) = \sum_{\hat{e}=0}^{n+l} P_{dp}(x, y, \hat{e}) P_{dif}^{s=r}(\hat{e}, e^{(s)}/2),\ P_{dn}^{(s)}(x, y, e^{(s)}) = \sum_{\hat{e}=0}^{n+l} P_{dn}(x, y, \hat{e}) P_{dif}^{s=r}(\hat{e}, e^{(s)}/2), \tag{4.7}$$

where $P_{dp}(x, y, e)$ and $P_{dn}(x, y, e)$ are the probabilities calculated from Section IV and $P_{dif}^{s=r}(\hat{e}, \hat{e}')$ is the probability that $\hat{D}(\boldsymbol{x}, \boldsymbol{y}) = \hat{e}'$ given that $D(\boldsymbol{x}, \boldsymbol{y}) = \hat{e}$ and $s = r$. We can calculate this probability by examining Table 1. We observe that each bit flip from 0 to 1, neglecting the fraction parts, will either decrease $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ by 1 or keep it the same. Also, each bit flip from 1 to 0, neglecting the fraction parts, will either increase $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ by 1 or keep it the same. $P_{dif}^{s=r}(\hat{e}, \hat{e}')$ can therefore be calculated as:

$$P_{dif}^{s=r}(\hat{e}, \hat{e}') = \sum_{s=|\hat{e}'-\hat{e}|}^{n+l} \sum_{w=|\hat{e}'-\hat{e}|}^{\min(s+|\hat{e}'-\hat{e}|, s)} \left[\binom{s}{w} (\frac{1}{2})^s \binom{s}{w- \mid \hat{e}' - \hat{e} \mid} (\frac{1}{2})^s \binom{n+l}{s} \binom{n+l}{s} p^{2s} (1-p)^{2n+2l-2s} \right]. \tag{4.8}$$

In the case that $e^{(s)}$ is an odd number, $e^{(s)}$ can arise from two cases: either $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ is on the left-hand-side of $(e^{(s)} - 1)/2$, i.e., $r < s$, or $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ is on the right-hand-side of $(e^{(s)} + 1)/2$, i.e., $r > s$. When $e^{(s)}$is odd, $P_{dp}^{(s)}(x, y, e)$ and $P_{dn}^{(s)}(x, y, e)$ can be expressed as:

$$P_{dp}^{(s)}(x, y, e^{(s)}) = \sum_{\hat{e}=0}^{n+l} \{ P_{dp}(x, y, \hat{e}) [P_{dif}^{s>r}(\hat{e}, (e^{(s)} - 1)/2) + P_{dif}^{s<r}(\hat{e}, (e^{(s)} + 1)/2)] \},$$

$$P_{dn}^{(s)}(x, y, e^{(s)}) = \sum_{\hat{e}=0}^{n+l} \{ P_{dn}(x, y, \hat{e}) [P_{dif}^{s>r}(\hat{e}, (e^{(s)} - 1)/2) + P_{dif}^{s<r}(\hat{e}, (e^{(s)} + 1)/2)] \}.$$

$P_{dif}^{s>r}(\hat{e}, \hat{e}')$ is defined to be the probability that $\text{nint}(\tilde{D}(\boldsymbol{x}, \boldsymbol{y})) = \hat{e}'$ given $D(\boldsymbol{x}, \boldsymbol{y}) = \hat{e}$ and $s > r$. $P_{dif}^{s<r}(\hat{e}, \hat{e}')$ is defined to be the probability that $\text{nint}(\tilde{D}(\boldsymbol{x}, \boldsymbol{y})) = \hat{e}'$ given $D(\boldsymbol{x}, \boldsymbol{y}) = \hat{e}$ and $s < r$.

From Table I, we can express $P_{dif}^{s>r}(\hat{e},\hat{e}')$ and $P_{dif}^{s>r}(\hat{e},\hat{e}')$ as following:

$$
\begin{aligned}
&P_{dif}^{s>r}(\hat{e},\hat{e}') \\
&= \mathbb{1}(\hat{e}>\hat{e}') \sum_{s=\hat{e}-\hat{e}'}^{n+l} \sum_{r=0}^{s-1} \sum_{w=\hat{e}-\hat{e}'}^{\min(r+\hat{e}-\hat{e}',s)} \left\{ \binom{s}{w} (\frac{1}{2})^{s+r} \binom{r}{w-\hat{e}+\hat{e}'} \binom{n+l}{s} \binom{n+l}{r} p^{s+r}(1-p)^{2n+2l-s-r} \right\}, \\
&+ \mathbb{1}(\hat{e}\leq\hat{e}') \sum_{r=\hat{e}-\hat{e}'}^{n+l} \sum_{s=r+1}^{n+l} \sum_{h=\hat{e}-\hat{e}'}^{\min(r+\hat{e}-\hat{e}',s)} \left\{ \binom{s}{h} (\frac{1}{2})^{s+r} \binom{r}{h-\hat{e}+\hat{e}'} \binom{n+l}{s} \binom{n+l}{r} p^{s+r}(1-p)^{2n+2l-s-r} \right\}
\end{aligned}
$$

$$
\begin{aligned}
&P_{dif}^{s<r}(\hat{e},\hat{e}') \\
&= \mathbb{1}(\hat{e}\geq\hat{e}') \sum_{s=\hat{e}-\hat{e}'}^{n+l} \sum_{r=s+1}^{n+l} \sum_{w=\hat{e}-\hat{e}'}^{\min(r+\hat{e}-\hat{e}',s)} \left\{ \binom{s}{w} (\frac{1}{2})^{s+r} \binom{r}{w-\hat{e}+\hat{e}'} \binom{n+l}{s} \binom{n+l}{r} p^{s+r}(1-p)^{2n+2l-s-r} \right\}, \\
&+ \mathbb{1}(\hat{e}<\hat{e}') \sum_{r=\hat{e}-\hat{e}'}^{n+l} \sum_{s=0}^{s-1} \sum_{h=\hat{e}-\hat{e}'}^{\min(r+\hat{e}-\hat{e}',s)} \left\{ \binom{s}{h} (\frac{1}{2})^{s+r} \binom{r}{h-\hat{e}+\hat{e}'} \binom{n+l}{s} \binom{n+l}{r} p^{s+r}(1-p)^{2n+2l-s-r} \right\}
\end{aligned}
$$

where $\mathbb{1}$ denotes the indicator function.

This ends our derivation for $P_{dp}^{(s)}(x,y,e^{(s)})$ and $P_{dn}^{(s)}(x,y,e^{(s)})$. These two probabilities can then be used in place of $P_{dp}(x,y,e)$ and $P_{dn}(x,y,e)$ from Section IV. We proceed with the rest of the calculation in Section IV using $e^{(s)}$ instead of e and calculate the classification accuracy accordingly.

4.5.2 Average-Case Study Results Using Soft Hamming Scheme

We use the above equations to calculate the expected accuracy of the k-NN classifier where the Hamming distance is computed using the Soft Hamming scheme on the 3-of-5/2 concept. We set $N = 32$ and the accuracy is selected to be the maximum accuracy for $2 \leq k \leq 16$. The attribute noise levels are characterized by the cross over probabilities of the write BSC, P_{bsc}.

We also plot the expected classification accuracy of the noise-free case and the expected classification accuracy under attribute noise without error-detection/error-correction. We observe that in the region where attribute noise is small, e.g., $p_{bsc} < 10^{-2}$, the attribute noise has little effect on classification accuracy. In the region where attribute noise is large, e.g.,

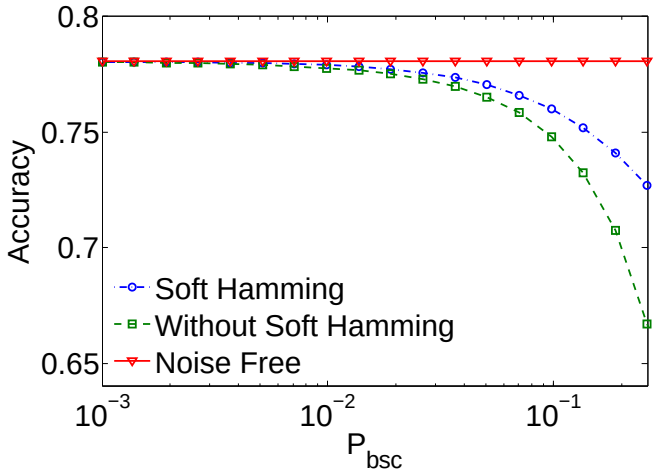


Figure 4.1: attribute noise v.s. max accuracy

$p_{bsc} > 10^{-2}$, our Soft Hamming scheme can successfully improve the classification accuracy under attribute noise.

CHAPTER 5

Error-Detection and Error-Correction under Resistance Variation and Write BSC

In previous chapters, we proposed HD-EIM to feasibly compute Hamming distance in resistive memory under the adverse effect of memristor variability due to resistance variation. We also provided the Soft Hamming scheme to deal with single bit error introduced by memristor variability due to nondeterministic switching mechanism. In this chapter, we seek to combine the two approaches in order to deal with memristor variability due to both sources. Again, we suppose $\boldsymbol{x}^{(c)}$ and $\boldsymbol{y}^{(c)}$ are inversion coded vectors to be stored. And we define $\tilde{\boldsymbol{x}}^{(c)}$ and $\tilde{\boldsymbol{y}}^{(c)}$ to be the noisy vectors actually stored due to write BSC. In order to make the math tractable, we provide a solution to a simpler problem, where $\sigma_H^2 = \sigma_L^2 = \sigma_0^2$.

With resistance variation taken into account, the normalized conductance measurement is a random variable that is additionally affected by the write BSC. Combining Equations (3.13) and (4.1), we have the distribution of the normalized conductance measurement as follows:

$$\tilde{G}(\tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)}) \sim \mathcal{N}\left(\tilde{N}'_{11} + (\tilde{N}'_{01} + \tilde{N}'_{10})2\epsilon + \tilde{N}'_{00}\epsilon, \frac{\tilde{N}'_{11}\sigma_0^2}{2\mu_H^2} + \frac{(\tilde{N}'_{01} + \tilde{N}'_{10})8\sigma_0^2}{2\mu_H^2} + \frac{\tilde{N}'_{00}\sigma_0^2}{2\mu_H^2}\right). \quad (5.1)$$

Here $\tilde{G}(\tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)})$ is a random variable denoting the normalized conductance measurement between rows of memristors that store $\tilde{\boldsymbol{x}}^{(c)}$ and $\tilde{\boldsymbol{y}}^{(c)}$. $\tilde{N}'_{gh}$ is defined to be the number of coordinates having bit g in $\boldsymbol{x}^{(c)}$ and bit h in $\boldsymbol{y}^{(c)}$, for $g, h \in \{0, 1\}$. We use the following

equation to calculate an intermediate random variable $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$:

$$\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{1-3\epsilon}[n + n\epsilon - \tilde{G}(\tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)})]. \tag{5.2}$$

Once again, there are four different fundamental types of errors for a pair of elements $x_i^{(c)}$ and $y_i^{(c)}$, $(0,0) \to (0,1)$, $(0,1) \to (0,0)$, $(0,1) \to (1,1)$ and $(1,1) \to (0,1)$, which we define them as error types A, B, C, and D, respectively.

Let us define a random variable E that represents the error type of a single bit error and whether the error occurs or not.

$$P(E) = \begin{cases} p_e, & \text{if} \quad E = A, B, C, D \\ 1 - 4p_e & \text{if} \quad E = 0, \end{cases}$$

where $E = A, B, C, D$ stands for the case that a single type A, B, C or D error occurs, respectively; $E = 0$ stands for the error-free case. We define p_e to be the probability that a single bit error occurs in a pair of coded vector $\boldsymbol{x}^{(c)}$ and $\boldsymbol{y}^{(c)}$, which can be calculated from the BSC parameter.

We then study the conditional distribution of $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$. Given $D(\boldsymbol{x}, \boldsymbol{y})$, for the error-free case, i.e., E = 0, $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ assumes the following distribution:

$$\mathcal{N}\left(D(\boldsymbol{x}, \boldsymbol{y}), \frac{(n + 7D(\boldsymbol{x}, \boldsymbol{y}))\sigma_0^2}{\mu_H^2(1-3\epsilon)^2}\right). \tag{5.3}$$

In the Table 5.1, we summarize the distribution of $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ for a single error of each type, in terms of its mean and variance. To simplify notation, we define $\gamma = \frac{\epsilon}{1-3\epsilon}$ and $\lambda = \frac{\sigma_0^2}{\mu_H^2(1-3\epsilon)^2}$. Table 5.1 is derived by examining (5.1) and Equation (5.2) for each error type.

From Table 5.1 and Equation (5.3), we have the conditional pdf, $f(\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) \mid D(\boldsymbol{x}, \boldsymbol{y}), E)$ for each pair of $D(\boldsymbol{x}, \boldsymbol{y}) \in \{0, ..., n\}$ and $E \in \{0, A, B, C, D\}$. We can therefore estimate E

Table 5.1: Fundamental Error types under Resistance Variation

Error Type	Mean	Variance
A	$D(\boldsymbol{x},\boldsymbol{y})-\gamma$	$\lambda(n+7D(\boldsymbol{x},\boldsymbol{y})+7/2)$
B	$D(\boldsymbol{x},\boldsymbol{y})+\gamma$	$\lambda(n+7D(\boldsymbol{x},\boldsymbol{y})-7/2)$
C	$D(\boldsymbol{x},\boldsymbol{y})-\gamma-1$	$\lambda(n+7D(\boldsymbol{x},\boldsymbol{y})-7/2)$
D	$D(\boldsymbol{x},\boldsymbol{y})+\gamma+1$	$\lambda(n+7D(\boldsymbol{x},\boldsymbol{y})+7/2)$

and $D(\boldsymbol{x},\boldsymbol{y})$ given the observation $\tilde{D}(\boldsymbol{x},\boldsymbol{y})$. Denoting our estimation of the pair $\{E, D(\boldsymbol{x},\boldsymbol{y})\}$ to be the pair $\{\hat{E}, \hat{D}(\boldsymbol{x},\boldsymbol{y})\}$, respectively, we have the following MAP (maximum *a posteriori* probability) estimator:

$$\begin{aligned}
\{\hat{E}, \hat{D}(\boldsymbol{x},\boldsymbol{y})\} &= \underset{\{E,D(\boldsymbol{x},\boldsymbol{y})\}}{\arg\max} \quad f(D(\boldsymbol{x},\boldsymbol{y}), E \mid \tilde{D}(\boldsymbol{x},\boldsymbol{y})) \\
&= \underset{\{E,D(\boldsymbol{x},\boldsymbol{y})\}}{\arg\max} \quad \frac{f(\tilde{D}(\boldsymbol{x},\boldsymbol{y}) \mid D(\boldsymbol{x},\boldsymbol{y}), E) f(D(\boldsymbol{x},\boldsymbol{y}), E)}{f(\tilde{D}(\boldsymbol{x},\boldsymbol{y}))} \\
&= \underset{\{E,D(\boldsymbol{x},\boldsymbol{y})\}}{\arg\max} \quad f(\tilde{D}(\boldsymbol{x},\boldsymbol{y}) \mid D(\boldsymbol{x},\boldsymbol{y}), E) f(D(\boldsymbol{x},\boldsymbol{y}), E) \qquad (5.4) \\
&= \underset{\{E,D(\boldsymbol{x},\boldsymbol{y})\}}{\arg\max} \quad f(\tilde{D}(\boldsymbol{x},\boldsymbol{y}) \mid D(\boldsymbol{x},\boldsymbol{y}), E) P(E) P(D(\boldsymbol{x},\boldsymbol{y})) \\
&= \underset{\{E,D(\boldsymbol{x},\boldsymbol{y})\}}{\arg\max} \quad f(\tilde{D}(\boldsymbol{x},\boldsymbol{y}) \mid D(\boldsymbol{x},\boldsymbol{y}), E) P(E).
\end{aligned}$$

Next, we present a solution to the above MAP estimator. Define $s_m(D_H)$ for $D_H \in \{0, ..., n-1\}$ to be a solution of equation:

$$f(\tilde{D}(\boldsymbol{x},\boldsymbol{y}) \mid D(\boldsymbol{x},\boldsymbol{y}) = D_H, E = B) = f(\tilde{D}(\boldsymbol{x},\boldsymbol{y}) \mid D(\boldsymbol{x},\boldsymbol{y}) = D_H + 1, E = A), \qquad (5.5)$$

such that $D_H + \gamma < s_m(D_H) < D_H + 1$.

Note that for Equation (5.5), there always exists a solution that satisfies the constraint of $s_m(D_H)$ therefore $s_m(D_H)$ always exist.

We define $s_0^{(1)}$ and $s_0^{(2)}$ to be solutions of the following equation:

$$f(\tilde{D}(\boldsymbol{x},\boldsymbol{y}) \mid D(\boldsymbol{x},\boldsymbol{y}) = 0, E = A) = f(\tilde{D}(\boldsymbol{x},\boldsymbol{y}) \mid D(\boldsymbol{x},\boldsymbol{y}) = 0, E = C), \tag{5.6}$$

such that:

$$-1-\gamma < s_0^{(1)} < -\gamma, \text{and } s_0^{(2)} < -1-\gamma.$$

We also define s_n to be a solution of the equation:

$$f(\tilde{D}(\boldsymbol{x},\boldsymbol{y}) \mid D(\boldsymbol{x},\boldsymbol{y}) = n, E = B) = f(\tilde{D}(\boldsymbol{x},\boldsymbol{y}) \mid D(\boldsymbol{x},\boldsymbol{y}) = n, E = D), \tag{5.7}$$

such that:

$$n+\gamma < s_n < n+1+\gamma.$$

We define $s_+(D_H)$ for $D_H \in \{0,...,n\}$ to be a solution, if exist, of the equation:

$$(1-4p_e) f(\tilde{D}(\boldsymbol{x},\boldsymbol{y}) \mid D(\boldsymbol{x},\boldsymbol{y}) = D_H, E = 0) = p_e f(\tilde{D}(\boldsymbol{x},\boldsymbol{y}) \mid D(\boldsymbol{x},\boldsymbol{y}) = D_H, E = B), \tag{5.8}$$

such that:

$$D_H < s_+(D_H) < s_m(D_H) \quad \text{for} \quad D_H \in \{0,...,n-1\},$$
$$n < s_+(D_H) < s_n \quad \text{for} \quad D_H = n.$$

Similarly, we define $s_-(D_H), D_H \in \{0,...,n\}$ to be a solution, if exist, of the equation:

$$(1-4p_e) f(\tilde{D}(\boldsymbol{x},\boldsymbol{y}) \mid D(\boldsymbol{x},\boldsymbol{y}) = D_H, E = 0) = p_e f(\tilde{D}(\boldsymbol{x},\boldsymbol{y}) \mid D(\boldsymbol{x},\boldsymbol{y}) = D_H, E = A), \tag{5.9}$$

such that:

$$s_m(D_H) < s_-(D_H) < D_H \quad \text{for} \quad D_H \in \{1,...,n\},$$
$$s_0^{(1)} < s_-(D_H) < 0 \quad \text{for} \quad D_H = 0.$$

Note that there exist cases where none of the solutions of Equation (5.8) or Equation (5.9) satisfies the constraint of $s_+(D_H)$ or $s_-(D_H)$. Hence $s_+(D_H)$ or $s_-(D_H)$ may not exist for certain device parameters.

If $s_+(D_H)$ and $s_-(D_H)$ exist for all D_H, and the following conditions are true:

$$\begin{aligned} &p_e f(s_m(D_H) \mid D(\boldsymbol{x}, \boldsymbol{y}) = D_H, E = B) = p_e f(s_m(D_H) \mid D(\boldsymbol{x}, \boldsymbol{y}) = D_H + 1, E = A) \\ \geq &(1 - 4p_e) f(s_m(D_H) \mid D(\boldsymbol{x}, \boldsymbol{y}) = D_H, E = 0), \quad \text{for} \quad D_H \in \{0, ..., n-1\}; \end{aligned}$$

and

$$\begin{aligned} &p_e f(s_m(D_H) \mid D(\boldsymbol{x}, \boldsymbol{y}) = D_H, E = B) = p_e f(s_m(D_H) \mid D(\boldsymbol{x}, \boldsymbol{y}) = D_H + 1, E = A) \\ \geq &(1 - 4p_e) f(s_m(D_H) \mid D(\boldsymbol{x}, \boldsymbol{y}) = D_H + 1, E = 0), \quad \text{for} \quad D_H \in \{0, ..., n-1\}; \end{aligned}$$

and

$$\begin{aligned} &p_e f(s_0^{(1)} \mid D(\boldsymbol{x}, \boldsymbol{y}) = 0, E = A) = p_e f(s_0^{(1)} \mid D(\boldsymbol{x}, \boldsymbol{y}) = 0, E = C) \\ \geq &(1 - 4p_e) f(s_0^{(1)} \mid D(\boldsymbol{x}, \boldsymbol{y}) = 0, E = 0); \end{aligned}$$

and

$$\begin{aligned} &p_e f(s_n \mid D(\boldsymbol{x}, \boldsymbol{y}) = n, E = B) = p_e f(s_n \mid D(\boldsymbol{x}, \boldsymbol{y}) = n, E = D) \\ \geq &(1 - 4p_e) f(s_n \mid D(\boldsymbol{x}, \boldsymbol{y}) = n, E = 0); \end{aligned}$$

then the MAP estimator has a solution as follows:

$$
\begin{aligned}
&\text{If } s_-(D_H) < \tilde{D}(\boldsymbol{x},\boldsymbol{y}) \le s_+(D_H) \quad \text{for some } D_H, \text{ then } \hat{E}=0, \hat{D}(\boldsymbol{x},\boldsymbol{y}) = D_H;\\
&\text{if } s_+(D_H) < \tilde{D}(\boldsymbol{x},\boldsymbol{y}) \le s_m(D_H) \quad \text{for some } D_H,\\
&\quad \text{then } \hat{E}=B, \hat{D}(\boldsymbol{x},\boldsymbol{y}) = D_H, \text{ or } \hat{E}=D, \hat{D}(\boldsymbol{x},\boldsymbol{y}) = D_H - 1;\\
&\text{if } s_m(D_H-1) < \tilde{D}(\boldsymbol{x},\boldsymbol{y}) \le s_-(D_H) \quad \text{for some } D_H,\\
&\quad \text{then } \hat{E}=A, \hat{D}(\boldsymbol{x},\boldsymbol{y}) = D_H, \text{ or } \hat{E}=C, \hat{D}(\boldsymbol{x},\boldsymbol{y}) = D_H + 1;\\
&\text{if } s_0^{(1)} < \tilde{D}(\boldsymbol{x},\boldsymbol{y}) \le s_-(0) \text{ then } \hat{E}=A, \hat{D}(\boldsymbol{x},\boldsymbol{y}) = 0, \text{or } \hat{E}=C, \hat{D}(\boldsymbol{x},\boldsymbol{y}) = 1;\\
&\text{if } s_+(n) < \tilde{D}(\boldsymbol{x},\boldsymbol{y}) \le s_n \text{ then } \hat{E}=B, \hat{D}(\boldsymbol{x},\boldsymbol{y}) = n, \text{or } \hat{E}=D, \hat{D}(\boldsymbol{x},\boldsymbol{y}) = n-1;\\
&\text{if } s_0^{(2)} < \tilde{D}(\boldsymbol{x},\boldsymbol{y}) \le s_0^{(1)}, \text{ then } \hat{E}=C, \hat{D}(\boldsymbol{x},\boldsymbol{y}) = 0;\\
&\text{if } \tilde{D}(\boldsymbol{x},\boldsymbol{y}) > s_n, \text{ then } \hat{E}=D, \hat{D}(\boldsymbol{x},\boldsymbol{y}) = n;\\
&\text{if } \tilde{D}(\boldsymbol{x},\boldsymbol{y}) \le s_0^{(2)}, \text{ then } \hat{E}=A, \hat{D}(\boldsymbol{x},\boldsymbol{y}) = 0, \text{or } \hat{E}=C, \hat{D}(\boldsymbol{x},\boldsymbol{y}) = 1.
\end{aligned}
\tag{5.10}
$$

Note that the MAP estimator has multiple solutions when an error is detected, i.e., $E \neq 0$. This is similar to what we observed in Section V and it is due to the underlying problem, e.g., $f(\tilde{D}(\boldsymbol{x},\boldsymbol{y})|D(\boldsymbol{x},\boldsymbol{y}) = D_H, E = B) = f(\tilde{D}(\boldsymbol{x},\boldsymbol{y})|D(\boldsymbol{x},\boldsymbol{y}) = D_H - 1, E = D)$. As a solution, we adapt the same idea of the Soft Hamming scheme and propose the following

estimator:

$$
\begin{aligned}
&\text{If } s_-(D_H) < \tilde{D}(\boldsymbol{x},\boldsymbol{y}) \le s_+(D_H) \quad \text{for some } D_H, \text{ then } \hat{D}(\boldsymbol{x},\boldsymbol{y}) = D_H;\\
&\text{if } s_+(D_H) < \tilde{D}(\boldsymbol{x},\boldsymbol{y}) \le s_m(D_H) \quad \text{for some } D_H, \text{ then } \hat{D}(\boldsymbol{x},\boldsymbol{y}) = D_H - 1/2;\\
&\text{if } s_m(D_H - 1) < \tilde{D}(\boldsymbol{x},\boldsymbol{y}) \le s_-(D_H) \quad \text{for some } D_H, \text{ then } \hat{D}(\boldsymbol{x},\boldsymbol{y}) = D_H + 1/2;\\
&\text{if } s_0^{(1)} < \tilde{D}(\boldsymbol{x},\boldsymbol{y}) \le s_-(0), \text{ then } \hat{D}(\boldsymbol{x},\boldsymbol{y}) = 1/2;\\
&\text{if } s_+(n) < \tilde{D}(\boldsymbol{x},\boldsymbol{y}) \le s_n, \text{ then } \hat{D}(\boldsymbol{x},\boldsymbol{y}) = n - 1/2;\\
&\text{if } s_0^{(2)} < \tilde{D}(\boldsymbol{x},\boldsymbol{y}) \le s_0^{(1)}, \text{ then } \hat{D}(\boldsymbol{x},\boldsymbol{y}) = 0;\\
&\text{if } \tilde{D}(\boldsymbol{x},\boldsymbol{y}) > s_n, \text{ then } \hat{D}(\boldsymbol{x},\boldsymbol{y}) = n;\\
&\text{if } \tilde{D}(\boldsymbol{x},\boldsymbol{y}) \le s_0^{(2)}, \text{ then } \hat{D}(\boldsymbol{x},\boldsymbol{y}) = n - \frac{1}{2}.
\end{aligned}
\tag{5.11}
$$

Note that we do not specify the error type in the above estimator. $\hat{D}(\boldsymbol{x},\boldsymbol{y})$ is estimated by taking the average between two candidate estimation in Estimator (5.10). This estimator therefore can successfully estimate the Hamming distance between vectors in resistive memory under the adverse effect of resistance variation while also have the capability to detect and correct a single bit error caused by the write BSC. Also note that the threshold values, e.g., $s_\pm(D_H)$, can be precomputed and stored in table to improve efficiency.

CHAPTER 6

Conclusion

6.1 Summary of Our Results

In this thesis, under the assumption of limited accessible information, we studied the feasibility of Hamming Distance Computation-in-Memory (HD-CIM) under two main (and complementary) sources of memristor variability. We use the optimal constant weight code that preserves Hamming distance, i.e., inversion coding, to provide *a priori* knowledge of vector weights. Analysis of the Hamming-distance estimation is provided when the resistances of memristors are modeled as Gaussian random variables. Our Soft Hamming scheme is proposed to detect and correct a single bit error introduced by the write BSC. The two schemes are evaluated for the k-NN classifier using the average-case study framework. These two schemes are combined at the end to tackle both resistance variation and non-deterministic write mechanism simultaneously.

6.2 Future Directions

Future research will focus on reliable HD-CIM suitable for vectors whose weights are partially known, e.g., within a known range. Codes that map a vector with arbitrary weight to a vector whose weight is within a known range while preserving Hamming distance will be studied to further reduce redundancy.

APPENDIX A

Approximation Justification

In Chapter 2, the following approximations are made:

$$\frac{H_{i,x}H_{i,y}}{H_{i,x}+H_{i,y}} \approx F_i \sim \mathcal{N}(\mu_H/2, \sigma_H^2/8), \tag{A.1}$$

$$\frac{L_{i,x}L_{i,y}}{L_{i,x}+L_{i,y}} \approx T_i \sim \mathcal{N}(\mu_L/2, \sigma_L^2/8), \tag{A.2}$$

$$\frac{H_{i,x}L_{i,y}}{H_{i,x}+L_{i,y}} \approx S_i \sim \mathcal{N}\left(\frac{\mu_L}{1+\epsilon}, \frac{\sigma_L^2}{(1+\epsilon)^4}\right), \tag{A.3}$$

$$\frac{L_{i,x}H_{i,y}}{L_{i,x}+H_{i,y}} \approx S_i \sim \mathcal{N}\left(\frac{\mu_L}{1+\epsilon}, \frac{\sigma_L^2}{(1+\epsilon)^4}\right). \tag{A.4}$$

Here we show that the approximated distributions are close to the original distributions by studying these approximations using data from a variety of memristor technology reported in the literature [CL11]. We model the reported resistance variations for 9 types of ReRAM devices. For each ReRAM device, let L and H denote the random variables for low state conductance and high state conductance, respectively. We assume L and H follow the following two Gaussian distributions respectively:

$$H \sim \mathcal{N}(\mu_H, \sigma_H^2), L \sim \mathcal{N}(\mu_L, \sigma_L^2). \tag{A.5}$$

For each ReRAM device, μ_H and μ_L are set to be the reciprocal value of its mean low-state resistance and high-state resistance, i.e. R_{avg}^L and R_{avg}^H, respectively. σ_H and σ_L are

calculated using the following rule:

$$\sigma_H = \frac{1}{2}\min\left(\frac{1}{R^H_{min}} - \frac{1}{R^H_{avg}}, \frac{1}{R^H_{avg}} - \frac{1}{R^H_{max}}\right), \sigma_L = \frac{1}{2}\min\left(\frac{1}{R^L_{min}} - \frac{1}{R^L_{avg}}, \frac{1}{R^L_{avg}} - \frac{1}{R^L_{max}}\right),$$

where R^L_{min} and R^L_{max} are the smallest and largest low-state resistance value reported, respectively. Similarly, R^H_{min} and R^H_{max} are the smallest and largest low-state resistance values reported. The resulting models for the 9 ReRAM devices are summarized in Table III. For

Table A.1: ReRAM Models and Bhattacharyya Distances

	μ_L	σ_L	μ_H	σ_H	ϵ	β	D_B^1	D_B^2	D_B^3	D_B^4
$TiOx$	1.0e-3	2.5e-4	2.5e-2	2.5e-3	4.0e-2	2.5e-2	6.5e-4	5.0e-3	8.3e-5	8.3e-5
$HfOx^{(1)}$	1.0e-3	2.1e-4	5.0e-3	8.3e-4	2.0e-1	3.4e-1	2.0e-3	3.5e-3	9.2e-4	9.2e-4
$AuZrOx^{(1)}$	3.3e-7	1.0e-7	1.4e-2	2.1e-3	2.3e-5	4.5e-2	1.5e-3	8.4e-3	3.2e-7	2.4e-7
$SrZrO3$	5.0e-7	8.3e-8	1.7e-3	3.3e-4	3.0e-4	8.0e-2	3.0e-3	2.0e-3	2.5e-7	2.5e-7
$CuGeSe$	1.7e-6	3.3e-7	3.3e-4	6.7e-5	5.0e-3	8.2e-2	3.0e-3	3.0e-3	1.5e-6	1.6e-6
$CoOx$	1.3e-5	3.8e-6	2.0e-4	3.8e-5	6.3e-2	1.1e-1	2.5e-3	7.7e-3	2.9e-4	3.0e-4
$HfOx^{(2)}$	1.3e-5	3.8e-9	1.0e-4	2.5e-5	1.3e-4	1.3e-1	5.0e-3	7.7e-3	2.0e-7	2.6e-7
$TiON$	1.7e-7	3.3e-8	5.0e-5	1.6e-5	3.3e-3	2.3e-1	9.7e-3	3.0e-3	9.0e-6	9.0e-6
$AuZrOx^{(2)}$	2.5e-8	6.3e-9	1.0e-5	2.5e-6	2.5e-3	1.3e-1	5.0e-3	5.0e-3	8.0e-7	9.5e-7

each ReRAM device, we calculate four discretized distributions on the left hand side of the approximation. We first generate 10^7 samples for $H_{i,x}$, $H_{i,y}$, $L_{i,x}$ and $L_{i,y}$ according to the (A.5). Then we calculate the discretized distributions for expressions on the left-hand-side of Approximation (A.1), (A.2), (A.3) and (A.4) by dividing all samples into 100 bins with equal bin width. We also calculate the discretized approximated distributions (the right-hand-side of Approximation (A.1), (A.2), (A.3) and (A.4) using the same bins. For each approximation in (A.1), (A.2), (A.3) and (A.4) we calculate the Bhattacharyya distance for each pair of

distributions using the following equations:

$$D_B(p,q) = -\ln(BC(p,q)), \tag{A.6}$$

where

$$BC(p,q) = \sum_{x \in X=1,..,100} \sqrt{p(x)q(x)}, \tag{A.7}$$

where p is the discretized original distribution and q is the discretized approximated distribution.

The four distance metrics for each ReRAM device, D_B^1, D_B^2, D_B^3 and D_B^4—the Bhattacharyya distance between the left distribution and the right distribution of approximations (A.1), (A.2), (A.3) and (A.4), respectively—are listed in Table III. Bhattacharyya distance close to zero means the two distribution are close. Thus we have showed our approximations are suitable for a variety of ReRAM devices which have a large range of resistance variation.

References

[BKL+05] I. Baek, D. Kim, M. Lee *et al.*, "Multi-layer cross-point binary oxide resistive memory (OxRRAM) for post-NAND storage application," in *Proc. Int. Electron Devices Meeting (IEDM) Tech. Dig.*, Washington, DC, Dec. 2005, pp. 750–753.

[CC15] Y. Cassuto and K. Crammer, "In-memory Hamming similarity computation in resistive arrays," in *Proc. IEEE Int. Symp. on Inf. Theory (ISIT)*, Hong Kong, June 2015, pp. 819–823.

[CL11] A. Chen and M.-R. Lin, "Variability of resistive switching memories and its impact on crossbar array performance," in *Proc. IEEE Rel. Physics Symp. (IRPS)*, Monterey, CA, April 2011, pp. MY–7.

[CLG+11] X. Cao, X. Li, X. Gao *et al.*, "All-ZnO-based transparent resistance random access memory device fully fabricated at room temperature," *Journal of Physics D: Applied Physics*, vol. 44, no. 25, p. 255104, 2011.

[CZ14] C. P. Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on Big Data," *Inf. Sciences*, vol. 275, pp. 314–347, Aug. 2014.

[HXN+15] S. Hamdioui, L. Xie, H. A. D. Nguyen *et al.*, "Memristor based computation-in-memory architecture for data-intensive applications," in *Proc. IEEE Des. Automation & Test in Europe Conf. & Exhibition (DATE)*, Grenoble, France, Mar. 2015, pp. 1718–1725.

[KD09] B. Kulis and T. Darrell, "Learning to hash with binary reconstructive embeddings," in *Proc. Neural Inf. Process. Syst. (NIPS)*, Vancouver, Canada, Dec. 2009, pp. 1042–1050.

[MRPC+11] G. Medeiros-Ribeiro, F. Perner, R. Carter *et al.*, "Lognormal switching times for titanium dioxide bipolar memristors: origin and resolution," *Nanotechnology*, vol. 22, no. 9, p. 095702, Jan. 2011.

[NPF12] M. Norouzi, A. Punjani, and D. J. Fleet, "Fast search in Hamming space with multi-index hashing," in *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, Providence, RI, July 2012, pp. 3108–3115.

[NXX12] D. Niu, Y. Xiao, and Y. Xie, "Low power memristor-based ReRAM design with error correcting code," in *Proc. IEEE Des. Automation Conf. Asia and South Pacific (ASP-DAC)*, Sydney, Australia, Jan./Feb. 2012, pp. 79–84.

[OY97] S. Okamoto and N. Yugami, "An average-case analysis of the k-nearest neighbor classifier for noisy domains," in *Proc. Int. Joint Conf. on Artificial Intel. (IJCAI)*, Nagoya, Aichi, Japan, Aug. 1997, pp. 238–245.

[PS92] M. J. Pazzani and W. Sarrett, "A framework for average case analysis of conjunctive learning algorithms," *Machine Learning*, vol. 9, no. 4, pp. 349–372, 1992.

[SSS+08] D. B. Strukov, G. S. Snider, D. R. Stewart *et al.*, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, May 2008.

[VRK+09] P. O. Vontobel, W. Robinett, P. J. Kuekes *et al.*, "Writing to and reading from a nano-scale crossbar memory based on memristors," *Nanotechnology*, vol. 20, no. 42, p. 425204, Sep. 2009.

[WLW+10] M. Wang, W. Luo, Y. Wang *et al.*, "A novel Cu x Si y O resistive memory in logic technology with excellent data retention and resistance distribution for embedded applications," in *Proc. IEEE Symp. on VLSI Technol. (VLSIT)*, Honolulu, HI, June 2010, pp. 89–90.

[YPQ+11] W. Yi, F. Perner, M. S. Qureshi *et al.*, "Feedback write scheme for memristive switching devices," *Appl. Phys. A: Materials Science & Processing*, vol. 102, no. 4, pp. 973–982, Jan. 2011.