UNIVERSITY OF CALIFORNIA SAN DIEGO

A Formal Perspective on Hyperdimensional Computing

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Computer Science

by

Anthony Hitchcock Thomas

2023

DEDICATION

To John.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would first like to thank my committee chairs and advisors Sanjoy Dasgupta and Tajana Rosing. Your guidance, patience, and input have been invaluable throughout my long journey to producing this dissertation. I also want to give special thanks to my other committee members, Kamalika Chaudhuri, Alexander Cloninger, and Tara Javidi, who have been fantastic resources for developing my understanding of the basic principles behind learning and representation. I am also grateful to my many friends and colleagues in the SEELab and at UCSD more broadly. I wish to give special thanks to Arman Khachiyan who has been a part of my academic journey for over a decade: your even-keeled perspective on life and research has smoothed out many bumps in the road, and I am grateful for your friendship and support. I am also particularly grateful to Geelon So: you are a source of inspiration for your clean and insightful approach to analysis as well as a fantastic climbing partner and friend. I am also particularly grateful to Paxon Frady, Denis Kleyko, Chris Kymm, Bruno Olshausen, and Fritz Sommer, from the Redwood Center at UC Berkeley, with whom I have had many wonderful conversations, and who have provided a refreshing perspective on the challenges of learning. I am very grateful to those who have funded this work, in particular the Semiconductor Research Corporation (SRC), and Intel (through the ML-Wins project). Chapter 3 contains work done while the author was a summer intern at Intel. This work was supported in part by CRISP, one of six centers in JUMP, an SRC program sponsored by DARPA, in part by an SRC-Global Research Collaboration grant, GRC TASK 3021.001, GRC TASK 2942.001, DARPA-PA-19-03-03 Agreement HR00112090036, and also NSF grants 1527034, 1730158, 1826967, 2100237, 2112167, 2052809, 2003279, 1830399, 1911095, 2028040, 2003279, and 1911095.

I also wish to acknowledge my (soon to be) husband John: you are kind, brilliant, and loving. Your constant support, excellent cooking, and willingness to endure long, rambling, discussions about esoteric topics have helped me through many difficult times and made the good ones even better. I also want to express my constant admiration for my sister Zora, who never ceases to inspire. And finally, I am, of course, most grateful to my parents Chris and Alicia, who have made this entire endeavor possible. I love you all very much.

Chapter 2 contains material from “A Theoretical Perspective on Hyperdimensional Computing,” by Anthony Thomas, Sanjoy Dasgupta, Tajana Rosing as it appeared in the Journal of Artificial Intelligence Research, 2021. The dissertation author was the primary investigator and author of this paper.

Chapter 3 contains material from “An Analysis of Hashing Architectures for Scalable Hyperdimensional Computing,” by Anthony Thomas, Behnam Khaleghi, Tianqi Zhang, Weihong Xu, Gopi Krishna-Jha, Sanjoy Dasgupta, Nilesh Jain, Nageen Himayat, Ravi Iyer, Tajana Rosing, as it has been submitted for publication in IEEE Transactions on Neural Networks and Learning Systems. The dissertation author was the primary investigator and author of this material.

Chapter 4 contains material from “A Formal Perspective on Learning with Vector Symbolic Architectures,” by Anthony Thomas, Sanjoy Dasgupta, Tara Javidi, Tajana Rosing, as it has been submitted for publication in the 2023 Conference on Neural Information Processing Systems, 2023. The dissertation author was the primary investigator and author of this material.

VITA

2013	Bachelor of Science, University of California, Berkeley
2019	Master of Science, University of California San Diego
2023	Doctor of Philosophy, University of California San Diego

PUBLICATIONS

Arman Khachiyan, Anthony Thomas, Huye Zhou, Gordon H Hanson, Alex Cloninger, Tajana Rosing, Amit Khandelwal "Using Neural Networks to Predict Micro-Spatial Economic Growth" American Economic Review: Insights, vol. 4, np. 4, pp 491-506, 2022.

Justin Morris, Hin Wai Lui, Kenneth Stewart, Behnam Khaleghi, Anthony Thomas, Thiago Marback, Baris Aksanli, Emre Neftci, and Tajana Rosing. "HyperSpike: HyperDimensional computing for more efficient and robust spiking neural networks." In 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 664-669. IEEE, 2022.

Alireza Amirshai, Anthony Thomas, Amir Aminifar, Tajana Rosing, and David Atienza. "M2D2: Maximum-Mean-Discrepancy Decoder for Temporal Localization of Epileptic Brain Activities." IEEE Journal of Biomedical and Health Informatics (2022).

Anthony Thomas, Sanjoy Dasgupta, Tajana Rosing. "A Theoretical Perspective on Hyperdimensional Computing" Journal of Artificial Intelligence Research, vol. 72, pp. 215-249, 2021

Fatemeh Asgarinejad, Anthony Thomas, and Tajana Rosing. "Detection of epileptic seizures from surface EEG using hyperdimensional computing." In 2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), pp. 536-540. IEEE, 2020.

Behnam Khaleghi, Sahand Salamat, Anthony Thomas, Fatemeh Asgarinejad, Yeseong Kim, and Tajana Rosing. "Shearer: highly-efficient hyperdimensional computing by software-hardware enabled multifold approximation." In Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design, pp. 241-246. 2020.

Anthony Thomas, Amir Aminifar, David Atienza. "Noise-resilient and interpretable epileptic seizure detection" 2020 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1-5, 2020, IEEE.

Justin Morris, Mohsen Imani, Samuel Bosch, Anthony Thomas, Helen Shu, and Tajana Rosing. “CompHD: Efficient hyperdimensional computing using model compression.” In 2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED), pp. 1-6. IEEE, 2019.

Anthony Thomas, Yunhui Guo, Yeseong Kim, Baris Aksanli, Arun Kumar, Tajana Rosing. “Hierarchical and distributed machine learning inference beyond the edge” 2019 IEEE 16th International Conference on Networking, Sensing and Control (ICNSC), pp. 18-23, 2019

Anthony Thomas and Arun Kumar. “A comparative evaluation of systems for scalable linear algebra-based analytics” Proceedings of the VLDB Endowment, vol. 11, no. 13, pp. 2168-2182, 2018, VLDB Endowment.

FIELDS OF STUDY

Major Field: Computer Science (Artificial Intelligence and Machine Learning)

ABSTRACT OF THE DISSERTATION

A Formal Perspective on Hyperdimensional Computing

by

Anthony Hitchcock Thomas

Doctor of Philosophy in Computer Science

University of California San Diego, 2023

Professor Tajana Rosing, Chair
Professor Sanjoy Dasgupta, Co-Chair

Hyperdimensional computing (HDC) is a paradigm, originating in the neuroscience literature, for computing on high-dimensional and distributed representations of data. The technique is simple to implement, amenable to formal analysis, and accords naturally with energy-efficient and highly parallel hardware platforms like FPGAs and “processing-in-memory” architectures. Indeed, recent years have seen substantial interest in leveraging the principles of HDC to develop highly efficient specialized hardware for cognitive information processing tasks. In HDC, all computation is performed on high-dimensional, distributed, representations of data, which are constructed using a variety of different encoding (i.e. embedding) techniques. In this dissertation, I develop a set of formal tools for analyzing the basic capabilities of these representations, and for obtaining sufficient conditions under which they can be used to effect various cognitive information processing tasks like learning and recall. I first develop a framework, based on the notion of incoherence popularized in the compressed sensing literature, for analyzing conditions under which specific data items, and collections thereof, can be embedded into HD-space in a manner that permits reliable recovery. I analyze the performance of these architectures in the presence of noise, and provide guidance on how to choose the dimension of the HD-space, a crucial consideration in practice. In applications of HDC, the encoding operation often represents a significant computational burden. This is particularly true when the input data is high-dimensional to begin with. I explore encoding architectures based on hashing that can mitigate these issues in certain settings, and provide novel analyses showing that they can offer substantial performance improvements over competing techniques, while enjoying similar formal guarantees for learning tasks. Implementation in hardware confirms these predictions. Finally, recent years have seen significant interest in using HDC as a substrate for learning tasks, in particular, classification. I develop a formal model for learning with HDC using techniques from statistical learning theory and kernel methods, an enormously successful approach to learning that also relies on unique properties of high-dimensional embeddings of data. This work clarifies the situations under which learning from HD representations will be successful, and elucidates the connections between HDC and related areas in classical machine learning. In contrast to prior work, that has also considered some of the questions introduced above, the analyses developed in this dissertation do not require asymptotic approximations, and readily yield formal results in the setting of finite-dimensional encodings one is limited to in practice.

Chapter 1

Introduction

Hyperdimensional computing (HDC) is an emerging area at the intersection of computer architecture, machine learning, and theoretical neuroscience [75]. It is based on the observation that brains are able to perform complex tasks using circuitry that: (1) uses low power, (2) requires low precision, and (3) is highly robust to data corruption. HDC aims to carry over similar design principles to a new generation of digital devices that are highly energy-efficient, fault tolerant, and well-suited to natural information processing [116].

The wealth of recent work on neural networks also draws its inspiration from the brain, but modern instantiations of these methods have diverged from the desiderata above. The success of these networks has rested upon choices that are not neurally plausible, most notably significant depth and training via backpropagation. Moreover, from a practical perspective, training these models often requires high precision and substantial amounts of energy. While a large body of literature has sought to ameliorate these issues with neural networks, these efforts have largely been designed to address specific performance limitations. By contrast, the properties above emerge naturally from the basic architecture of HDC.

Hyperdimensional computing focuses on the very simplest neural architectures. Typically, there is a single, static, mapping from inputs x to much higher-dimensional "neural" representations $\phi(x)$ living in some space $\mathscr{H}$. All computational tasks are performed in $\mathscr{H}$-space, using simple operations like element-wise addition, multiplication, and dot products. The mapping ϕ

is often taken to be random, and the embeddings have coordinates that have low precision; for instance, they might take values -1 and $+1$. The entire setup is elementary and lends itself to fast, low-power hardware realizations.

Indeed, recent years have seen substantial interest in developing optimized implementations of HDC-based algorithms on hardware accelerators [67, 116, 55, 128, 127, 68]. Broadly speaking, this line of work touts HDC as an energy efficient, low-latency, and noise-resilient alternative to conventional realizations of general purpose ML algorithms like support vector machines, multilayer perceptrons, and nearest-neighbor classifiers [93].

The basic ideas behind HDC have a long history in the cognitive-neuroscience literature [72, 106, 105, 138, 75], where they emerged as a mathematical model to describe neural information representation and symbolic reasoning. These ideas were subsequently extended to support representation of Euclidean data [110, 112] and for use in learning applications [85, 47]. This thesis is devoted to developing a formal understanding of the basic principles at play in these techniques. In particular, I address the following questions:

(1) How can individual pieces of data and collections thereof be represented in a manner that permits reliable recovery even in the presence of noise?

(2) What kinds of structure in the input ($\mathscr{X}$) space are preserved by the mapping to ϕ-space?

(3) How do choices about the HD architecture, like the choice of encoding function, dimension, and precision, affect the ability to learn from HD representations?

(4) What is the relationship between HDC and other techniques from machine learning?

(5) What are computationally efficient realizations of encoding algorithms that can scale to the setting that the raw data is, itself, high-dimensional?

Some of these questions have been introduced in prior work and studied in isolation or via informal arguments [105, 53, 85, 48, 49]. In this work I address these questions formally (e.g.

via mathematical-proof) and at a level of generality that allows one to abstract away from specific design choices like the precise construction of ϕ and $\mathscr{H}$.

In the remainder of this chapter, I provide an overview of HDC and its antecedents in the neuroscience literature. I then articulate what I see as the primary contributions of my work.

1.1 Introduction to Hyperdimensional Computing

1.1.1 Distributed Representations in Neuroscience

Neuroscience has proven to be a rich source of inspiration for the machine learning community: from the perceptron [122], which introduced a simple and general-purpose learning algorithm for linear classifiers, to neural networks [125], to convolutional architectures inspired by visual cortex [52], to sparse coding [100] and independent component analysis [12]. One of the most consequential discoveries from the neuroscience community, underlying much research at the intersection of neuroscience and machine learning, has been the notion of *high-dimensional distributed representations* as the fundamental data structure for diverse types of information. In the neuroscience context, these representations are also typically *sparse*.

To give a concrete example, the sensory systems of many organisms have a critical component consisting of a transformation from relatively low dimensional sensory inputs to much higher-dimensional *sparse* representations. These latter representations are then used for subsequent tasks such as recall and learning. In the olfactory system of the fruit fly [91, 146, 154, 21], the mapping consists of two steps that can be roughly captured as follows:

1. An input $x \in \mathbb{R}^n$ is collected via a sensory organ and mapped under a *random linear transformation* to a point $\phi(x) \in \mathbb{R}^d$ ($d \gg n$) in a high-dimensional space.

2. The coordinates of $\phi(x)$ are "sparsified" by a thresholding operation that just retains the locations of the largest k coordinates.

In the fly, the olfactory input is a roughly 50-dimensional vector ($n = 50$) corresponding to different types of odor receptor neurons while the sparse representation to which it is mapped is roughly 2,000-dimensional ($d = 2000$). A similar "expand-and-sparsify" template is also found in other species, suggesting that this process somehow exposes the information present in the input signal in a way that is amenable to learning by the brain [141, 101, 24]. The precise mechanisms by which this occurs are still not fully understood, but may have close connections to some of the literature on the theory of neural networks and kernel methods [35, 9, 119].

1.1.2 HD Computing

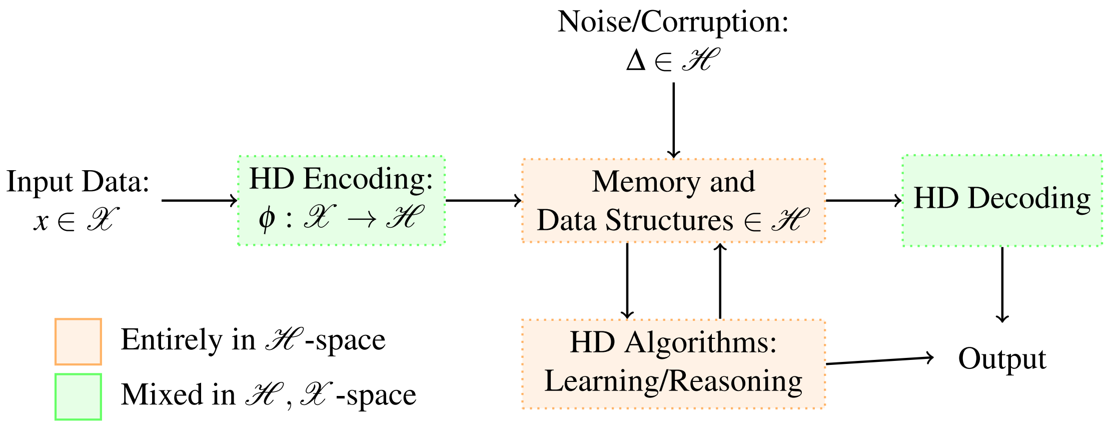

Noise/Corruption: $\Delta \in \mathscr{H}$
Input Data: $x \in \mathscr{X}$
HD Encoding: $\phi : \mathscr{X} \to \mathscr{H}$
Memory and Data Structures $\in \mathscr{H}$
HD Decoding
HD Algorithms: Learning/Reasoning
Output
Entirely in $\mathscr{H}$-space
Mixed in $\mathscr{H}$, $\mathscr{X}$-space

Figure 1.1. The flow of data in HD computing. Data is mapped from the input space to HD-space under an encoding function $\phi : \mathscr{X} \to \mathscr{H}$. HD representations of data are stored in data structures and may be corrupted by noise or hardware failures. HD representations can be used as input for learning algorithms or other information processing tasks and may be decoded to recover the input data.

The notion of high-dimensional and distributed data representations has engendered a number of computational models that have collectively come to be known as hyperdimensional computing [75, 83]. There are many different HDC architectures in the literature, but all provide a mechanism to generate and manipulate high-dimensional distributed representations of data. The specific choice of operators used for these tasks is usually called a "vector-symbolic architecture" (VSA) [54]. Notable examples of VSAs include "holographic reduced representations" [106, 105], "binary spatter codes" [73, 74], and "matrix binding of additive terms" [53].

An overview of data-flow in HD computing is given in Figure 1.1. The first step in HD computing is encoding, which maps a piece of input data to its high-dimensional representation under some function $\phi : \mathscr{X} \to \mathscr{H}$. The nature of ϕ depends on the type of input and the choice of $\mathscr{H}$, and we will subsequently discuss and analyze a wide variety of encoding functions that are applicable to sets, sequences, structures, and Euclidean data. The space $\mathscr{H}$ is some d-dimensional real inner-product space. Work in the literature on HD computing has also explored complex-valued embeddings [105, 47], but we will restrict attention to the more common real-valued case. For computational reasons, it is common in practice to impose constraints on the precision of points in $\mathscr{H}$. We emphasize that the dimension of $\mathscr{H}$ need not, in general, be greater than that of $\mathscr{X}$. Indeed, it is often the case that encoding can actually reduce the dimension of the data without adversely effecting performance on downstream tasks.

The HD representations of data can be manipulated using simple element-wise operators. Two important such operations are "bundling" and "binding." The bundling operator is used to compile a set of elements in $\mathscr{H}$ and takes the form of a function $\oplus : \mathscr{H} \times \mathscr{H} \to \mathscr{H}$. The function takes two points in $\mathscr{H}$ and returns a third point that is similar to both operands. The binding operator is used to create tuples of points in $\mathscr{H}$ and is likewise a function $\otimes : \mathscr{H} \times \mathscr{H} \to \mathscr{H}$. The function takes a pair of points in $\mathscr{H}$ as input, and produces a third point dissimilar to both operands. We will make these definitions more precise subsequently.

Given the HD representation $\phi(\mathscr{S})$ of a set of items $\mathscr{S} \subset \mathscr{X}$ (produced by bundling the items), we may be interested to query the representation to determine if it contains the encoding of some $x \in \mathscr{X}$. To do so, we compute some similarity score $\rho(\phi(x), \phi(\mathscr{S}))$ and declare that the item is present in $\mathscr{S}$ if the similarity is greater than some critical value. This process can be used to decode the HD representation so as to recover the original points in $\mathscr{X}$ [105, 48, 75]. We may additionally wish to assert that we can decode reliably even if $\phi(\mathscr{S})$ has been corrupted by some noise process. We will mathematically characterize sufficient conditions for robust decoding under different noise models and input data types.

Beyond simply storing and recalling specific patterns, HD representations may also be used for learning. HD computing is most naturally applicable to classification problems. We will repeatedly return to the following basic template for classification with HDC throughout this work. Suppose we are given some collection of labeled examples $\mathscr{D} = \{(x_i, y_i)\}_{i=1}^{n}$, where $x_i \in \mathscr{X}$ and $y_i \in \{c_i\}_{i=1}^{k}$ is a categorical variable indicating the class label of a particular x_i. One simple form of HD classification bundles together the data so as to generate a "prototypical" example for the class [75, 85, 116]:

$$\theta_j = \bigoplus_{i=1}^{n} \alpha_{ij} \phi(x_i), \tag{1.1}$$

where α_{ij} is a weight assigned to the i-th example for the j-th class. The resulting θs are sometimes quantized to lower precision or sparsified via a thresholding operation. Common instantiations of this method are to simply bundle together the data for a particular class, in which case $\alpha_{ij} = \mathbb{1}(y_i = c_j)$ [116, 85], and to run the Perceptron algorithm [122], in which case $\oplus$ is the element-wise sum, and the weights are, after the first pass over the data, $\alpha_{ij} \in \{+1, 0, -1\}$, depending on how the algorithm made a mistake on the i-th example [62, 64]. Given some subsequent piece of query data $x_q \in \mathscr{X}$ for which we do not know the correct label, we simply return the label of the most similar prototype:

$$k^\star = \operatorname*{argmax}_{j \in 1,\ldots,k} \rho(\phi(x_q), \theta_j).$$

The similarity function ρ is typically taken to be the dot-product, with the operands normalized in some fashion if necessary. In Chapter 4, we will consider this general approach to learning in significantly greater detail, and consider properties of the HD encoding that can make linear models more powerful in HD space than in the original space.

HD computing and closely related techniques have been applied to a wide variety of practical problems in fields ranging from bio-signal processing [114, 62], to natural language processing [126], and robotics [94, 97]. This dissertation will be mostly concerned with a more abstract treatment that focuses on the basic properties of HD computing. The interested reader is referred to [114, 85, 83] for discussions related to practical aspects of HD computing.

1.2 Contribution of this Dissertation

The following section outlines each main chapter in this dissertation and highlights their key contributions. Broadly speaking, this dissertation has two goals. The first is predominantly theoretical in nature, and is to develop a mathematical toolkit for analyzing HDC and understanding its basic capabilities. In this regard, an important contribution of this work is to develop the analysis of HDC in the *finite-dimensional* setting one is limited to in practice. Most existing work has analyzed these architectures using properties of the encoding that hold in expectation, under specific assumptions on the encoding procedure, or by using asymptotic approximations (e.g. limit-theorems) [105, 53, 48, 47]. A key contribution of this work has been to develop analyses of HDC that both hold in significant generality, and do not require invoking limit-theorems or other asymptotic approximations. The basic techniques used to do so are foundational in computer science and machine learning, but have not been widely applied previously in the literature on HDC.

The second goal is more practical in nature and is to develop a more formal understanding of how the choice of key parameters like the encoding-dimension, precision, and (sometimes) sparsity effect the performance of various tasks, in particular, memory and learning. This is done by applying the theory described above to obtain bounds on these quantities that are sufficient for various tasks of interest to succeed. This highlights the importance of obtaining non-asymptotic results: in practice, one is limited to finite-dimensional representations, and attempting to obtain rigorous guarantees in the finite-dimensional setting via asymptotic approximations is a potentially fraught undertaking that requires significant care. By contrast, the techniques used in the analysis here directly yield such results.

1.2.1 Chapter 2: Theoretical Foundations of HDC

The recent literature has suggested a variety of different HD architectures that conform to the overall blueprint described above, but differ in many of their details. These architectures have historically been analyzed independently or using informal arguments (e.g. empirically or via asymptotic approximations) [105, 152, 48]. This chapter develops a novel mathematical framework that allows one to analyze different architectures in the finite-dimensional setting, and at a level of abstraction that allows their properties to be easily compared. This chapter unifies a wide range of architecture for encoding and decoding discrete data under the framework of representation in an incoherent dictionary, which formalizes the notion of "almost-orthogonal" codes that is encountered in earlier literature [105, 75]. This framework immediately yields non-asymptotic sufficient conditions under which a variety of encoding and decoding architectures will succeed, even in the presence of noise.

From a practical standpoint, this analysis allows one to obtain closed-form expressions that relate the choice of encoding dimension, properties of the underlying data, and noise to the probability of correct decoding, allowing practitioners to choose the encoding dimension (a key consideration in practice) in a principled fashion. The chapter concludes by discussing some methods for encoding Euclidean data and some preliminary implications for the use of HDC in learning applications, which are developed in significantly greater generality in Chapter 4.

1.2.2 Chapter 3: HDC and Hashing

Many of the encoding schemes studied in the previous sections conform to a common template. One views the input x as an m-dimensional vector, and generates an embedding via $\phi(x) = f(Mx)$, where $M \in \mathbb{R}^{d \times m}$ is a random-matrix used to embed the data into HD space, and f is some optional non-linearity applied element-wise to Mx. While simple, and remarkably powerful theoretically, basic instantiations of this approach encounter serious limitations when the ambient dimension (m) of the data is large, for the simple reason that one must store M and access it to look-up values. This chapter explores techniques from the literature on streaming algorithms [14, 71] for generating encodings “on-the-fly” using hashing. While prior work has noted the connections between these techniques and HDC [143, 84], they have not been explored for learning applications, and existing analyses do not substantiate their use in this setting. Analysis in this chapter rectifies these issues.

Using the formal model introduced in the previous chapter, it is shown that hash-based methods enjoy similar theoretical guarantees, in a learning setting, to standard techniques based on random codeword generation, while being substantially more efficient. New results are developed to analyze the tradeoff between the encoding dimension, the number and type of hash-functions, and intrinsic properties of the data like the size of the input domain, and provide sufficient conditions under which an important family of learning algorithms run on HD representations will succeed. These techniques are implemented in an FPGA and in-memory architecture,[1] and evaluated on a large scale classification problem [34], validating the theory and showing that hash-based encodings offer comparable (or superior) levels of accuracy to existing HDC techniques while being over substantially faster than the most competitive existing approaches in the HD literature. More broadly, this work opens a potentially fruitful line of connection between the hardware and implementation focused literature on HDC, and the theoretical literature on streaming algorithms, which makes heavy use of hashing based techniques [32, 71, 14].

1.2.3 Chapter 4: Statistical Learning with HDC

The bulk of formal work on HDC has focused on quantifying the ability of HD architectures to store and recall specific patterns–that is to say, on the problem of memory [105, 48, 144, 29]. However, recent years have seen significant interest in using the HD representations for learning tasks like classification and regression [85, 19, 62, 116]. Although there is some overlap between the basic methods used for both memory and learning, the goals are

[1]hardware work is done in collaboration with others

different. In the learning setting, one is typically interested in developing a model that can be used to predict some outcome of interest about new data that has not previously been seen, and the classic theory of storage capacity is not generally adequate for addressing this setting.

While the previous two chapters included some treatment of learning problems, this was limited to learning linear separators in HD space, and assumed the data was separable in its ambient representation, which is restrictive in practice. Chapter 4 develops a significantly more detailed analysis of learning from HD representations, and casts this problem in a more traditional framework of statistical learning and empirical-risk-minimization [149, 10]. The goal of this chapter is to bridge the gap between the classic capacity theory of HDC, which focuses on storing and retrieving specific data items, and the growing practical interest in learning from these representations. To make progress in this direction, this chapter provides a formal analysis of learning with HDC using techniques from the literature on kernel methods and statistical learning theory.

The chapter extend the capacity theory of HDC (e.g. many of the results from Chapter 2) to address learning settings, and develops a model, using techniques from statistical learning theory, that allows one to formally analyze the relationship between encoding dimension, precision, the number of samples presented to the learning algorithm, and performance on a particular task. This is the first work to address these questions in the context of HDC.

Finally, this chapter provides a more thorough accounting than is available in prior work, of the connections between HDC and kernel methods, a diverse and influential family of techniques from statistics and machine learning that also relies on learning from high-dimensional embeddings of data [135, 60]. In particular, it is shown that every encoding function induces a kernel on the data that can be used to characterize the family of learnable models, and the solution of minimum empirical-loss to a wide variety of learning problems posed on the VSA representations can be represented as a kernel machine, that is as a weighted linear combination of encodings of training data.

Chapter 2

Theoretical Foundations of HDC

A central object in HD computing is the mapping from inputs to their high-dimensional representations. The design of this mapping, typically referred to as "encoding" in the literature on HD computing, has been the subject of considerable research. There is a wide range of possible encoding methods. Some of these have been introduced in the HD computing literature and studied in isolation [105, 53, 85]. In this chapter, we present a novel unifying framework in which to study these mappings and to characterize their key properties in a non-asymptotic setting. In particular, we are concerned with the following questions introduced in the introduction:

(1) How can individual pieces of data and collections thereof be represented in a manner that permits reliable recovery, even in the presence of noise?

(2) What kinds of structure in the input space are preserved by the mapping to HD-space?

We focus first on encoding and decoding discrete data, and then turn to methods for representing data in a Euclidean space. We conclude the chapter by discussing the ability of encoding methods to preserve cluster structure and linear separability in the input space.

2.1 Encoding and Decoding Discrete Data

We begin our discussion of discrete data by focusing on encoding and decoding *sets*. We devote a considerable amount of time to this problem as many procedures for encoding more complex data types such as sequences essentially amount to transforming the data into a set and then applying the standard set encoding method.

2.1.1 Finite Sets

Let $\mathscr{A} = \{a_i\}_{i=1}^{m}$ be some finite alphabet of m symbols. Symbols $a \in \mathscr{A}$ are mapped to $\mathscr{H}$ under an encoding function $\phi : \mathscr{A} \to \mathscr{H}$. Our goal in this section is to consider the encoding of sets $\mathscr{S}$ whose elements are drawn from $\mathscr{A}$. The HD representation of $\mathscr{S}$ is constructed by superimposing the embeddings of the constituent elements using the bundling operator $\oplus : \mathscr{H} \times \mathscr{H} \to \mathscr{H}$. The encoding of $\mathscr{S}$ is defined to be $\phi(\mathscr{S}) = \oplus_{a \in \mathscr{S}} \phi(a)$. We first focus on the intuitive setting in which $\oplus$ is the element-wise sum and then address other forms of bundling.

To determine if some $a \in \mathscr{A}$ is contained in $\mathscr{S}$, we check if the dot product $\langle \phi(a), \phi(\mathscr{S}) \rangle$ exceeds some fixed threshold. If the codewords $\{\phi(a) : a \in \mathscr{A}\}$ are orthogonal and have a constant length L, then we have $\langle \phi(a), \phi(\mathscr{S}) \rangle = L^2\, \mathbb{1}(a \in \mathscr{S})$, where $\mathbb{1}$ is the indicator function which evaluates to one if its argument is true and zero otherwise. However, when the codewords are not perfectly orthogonal, we have $\langle \phi(a), \phi(\mathscr{S}) \rangle = L\mathbb{1}(a \in \mathscr{S}) + \Delta$, where Δ is the "cross-talk" caused by interference between the codewords. In order to decode reliably, we must ensure the contribution of the cross-talk is small and bounded. We formalize this using the notion of incoherence popularized in the sparse coding literature. We define incoherence formally as [42]:

Definition 1. ***Incoherence***. *For $\mu \geq 0$, we say $\phi : \mathscr{A} \to \mathscr{H}$ is μ-incoherent if for all distinct $a, a' \in \mathscr{A}$, we have*

$$|\langle \phi(a), \phi(a') \rangle| \leq \mu L^2$$

where $L = \min_{a \in \mathscr{A}} \|\phi(a)\|$.

When $d \geq m$, it is possible to have codewords that are mutually orthogonal, whereupon $\mu = 0$. In general, we will be interested in results that do not require $d \geq m$.

Exact Decoding of Sets

In the following section, we show how the cross-talk can be bounded in terms of the incoherence of ϕ, and use this to derive a simple threshold rule for exact decoding.

Theorem 1. *Let $L = \min_{a \in \mathscr{A}} \|\phi(a)\|$ and let the bundling operator be the element wise sum. To decode whether an element a lies in set S, we use the rule*

$$\langle \phi(a), \phi(S) \rangle \geq \frac{1}{2} L^2.$$

This gives perfect decoding for sets of size $\leq s$ if ϕ is $1/(2s)$-incoherent.

Proof. Consider some symbol a. Then:

$$\langle \phi(a), \phi(\mathscr{S}) \rangle = \mathbb{1}(a \in \mathscr{S}) \langle \phi(a), \phi(a) \rangle + \sum_{a' \in \mathscr{S} \setminus \{a\}} \langle \phi(a), \phi(a') \rangle$$

If $a \in \mathscr{S}$, then the above is lower bounded by $L^2 - sL^2\mu$, where μ is the incoherence of ϕ. Otherwise, it is upper bounded by $sL^2\mu$. So we decode perfectly if $sL^2\mu < L^2/2$, or $\mu < 1/(2s)$. □

Random Codebooks

In practice, each $\phi(a)$ is usually generated by sampling from some distribution over $\mathscr{H}$ or a subset thereof [75, 85, 116]. One typically requires that this distribution is factorized so that coordinates of $\phi(a)$ are independent and identically distributed. Intuitively, the incoherence condition stipulated in Theorem 1 will hold if dot products between two different codewords are concentrated around zero. Furthermore, we would like it to be the case that this concentration occurs quickly as the encoding dimension is increased. It turns out that a fairly broad family of simple distributions satisfies these properties.

As an example, suppose $\phi(a)$ is sampled from the uniform distribution over $\{\pm 1\}^d$, which we denote $\phi(a) \sim \{\pm 1\}^d$. In this case, $L = \sqrt{d}$ exactly, and a direct application of

Hoeffding's inequality and the union bound yields:

$$\mathbb{P}(\exists \text{ distinct } a, a' \in \mathscr{A} \text{ s.t. } |\langle \phi(a), \phi(a') \rangle| \geq \mu d) \leq m^2 \exp\left(-\frac{\mu^2 d}{2}\right).$$

(Recall that $m = |\mathscr{A}|$.) Stated another way, with high probability $\mu = O(\sqrt{(\ln m)/d})$, meaning that we can make μ as small as desired by increasing d.

In fact, the same basic approach holds for the much broader class of *sub-Gaussian* distributions, which can be characterized as follows [151]:

Definition 2. ***Sub-Gaussian Random Variable****. A random variable $X \sim P_X$ is said to be sub-Gaussian if there exists $\sigma \in \mathbb{R}^+$, referred to as the sub-Gaussian parameter, such that:*

$$\mathbb{E}[\exp(\lambda(X - \mathbb{E}[X]))] \leq \exp\left(\frac{\sigma^2 \lambda^2}{2}\right) \text{ for all } \lambda \in \mathbb{R}.$$

Intuitively, the tails of a sub-Gaussian random variable decay at least as fast those of a Gaussian. We say the encoding ϕ is σ-sub-Gaussian if $\phi(a)$ is generated by sampling its d coordinates independently from the same sub-Gaussian distribution with parameter σ. We say ϕ is "centered" if the distribution from which it is sampled is of mean zero. In general, we assume ϕ is centered unless stated otherwise.

Codewords drawn from a sub-Gaussian distribution have the useful property that their lengths concentrate fairly rapidly around their expected value. This concentration is, in general, worse than sub-Gaussian but well behaved nonetheless. The following result is well known but we reiterate it here as it is useful for our subsequent discussion.

Theorem 2. *Let ϕ be centered and σ-sub-Gaussian. Then:*

$$\mathbb{P}(\exists a \in \mathscr{A} \text{ s.t. } |\|\phi(a)\|_2^2 - \mathbb{E}[\|\phi(a)\|_2^2]| \geq t) \leq 2m \exp\left(-c \min\left\{\frac{t^2}{d\sigma^4}, \frac{t}{\sigma^2}\right\}\right)$$

for some positive absolute constant c.

Proof. The result is an immediate consequence of the Hanson-Wright inequality [56, 123] which holds that, for x a centered, d-dimensional, σ-sub-Gaussian random vector, and $A \in \mathbb{R}^{d\times d}$ an arbitrary square matrix, the quadratic form $x^T Ax$ obeys the following concentration bound:

$$\mathbb{P}(|x^T Ax - \mathbb{E}[x^T Ax]| \geq t) \leq 2\exp\left(-c\min\left(\frac{t^2}{\sigma^4\|A\|_F^2}, \frac{t}{\sigma^2\|A\|}\right)\right)$$

where c is a positive absolute constant, $\|A\|_F^2 = \sum_{i,j}|A_{ij}|^2$ is the Frobenius norm and $\|A\| = \max_{\|x\|\leq 1}\|Ax\|$ is the operator norm. The result follows by taking A to be the $d \times d$ identity matrix, in which case $x^T I_d x = \|x\|_2^2$, and union bounding over all m symbols in the alphabet. $\square$

Like the conventional Gaussian distribution, sub-Gaussianity is preserved under linear transformations. That is, if $x = \{x_i\}_{i=1}^n$ is a sequence of i.i.d. sub-Gaussian random variables and a is an arbitrary vector in $\mathbb{R}^n$, then $\langle a, x\rangle$ is sub-Gaussian with parameter $\sigma\|a\|_2$ [151]. We can obtain a more general version of the previous result about $\phi \sim \{\pm 1\}^d$ which applies to $\phi(a)$ sampled from any sub-Gaussian distribution.

Theorem 3. *Let ϕ be σ-sub-Gaussian. Then, for $\mu > 0$,*

$$\mathbb{P}(\exists \text{distinct } a, a' \in \mathscr{A} \text{ s.t. } |\langle\phi(a), \phi(a')\rangle| \geq \mu L^2) \leq m^2\exp\left(-\frac{\mu^2\kappa L^2}{2\sigma^2}\right)$$

where $\kappa = (\min_a\|\phi(a)\|^2)/(\max_a\|\phi(a)\|^2)$.

Proof. Fix some a and a'. Treating $\phi(a)$ as a fixed vector in $\mathbb{R}^d$ and using the fact that sub-Gaussianity is preserved under linear transformations, we may apply a Chernoff bound for sub-Gaussian random variables (e.g. Prop 2.1 of [151]) to obtain:

$$\mathbb{P}(|\langle\phi(a), \phi(a')\rangle| \geq \mu L^2) \leq 2\exp\left(-\frac{\mu^2 L^4}{2\sigma^2\|\phi(a)\|_2^2}\right) \leq 2\exp\left(-\frac{\mu^2 L^4}{2\sigma^2 L_{\max}^2}\right)$$

where $L_{\max} = \max_{a\in\mathscr{A}} \|\phi(a)\|_2$. Therefore, taking $\kappa = L^2/L_{\max}^2$, we have:

$$\mathbb{P}(|\langle\phi(a),\phi(a')\rangle| \geq \mu L^2) \leq 2\exp\left(-\frac{\mu^2\kappa L^2}{2\sigma^2}\right)$$

and the claim follows by applying the union bound over all $\binom{m}{2} < m^2/2$ pairs of codewords. We note that, per Theorem 2, $\kappa \to 1$ as d becomes large. $\square$

To be concrete and provide useful practical guidance, we here introduce three running examples of codeword distributions.

Dense Binary Codewords. In our first example, the most common in practice in our impression, $\phi(a)$ is sampled from the uniform distribution over the d-dimensional unit cube $\{-1,+1\}^d$. This approach is advantageous because it leads to efficient hardware implementations [62, 116] and is simple to analyze.

Gaussian Codewords. Our second example consists of codewords sampled from the d-dimensional Gaussian distribution [105]. That is, $\phi(a) \sim \mathscr{N}(0_d, \sigma^2 I_d)$, where 0_d is the d-dimensional zero vector. Here, the codewords will not be of exactly the same length. However, Theorem 2 ensures that squared codeword lengths are concentrated around their expected value of $\sigma^2 d$. More formally, for some $\tau > 0$:

$$\mathbb{P}(\exists a \in \mathscr{A} \text{ s.t. } |\|\phi(a)\|_2^2 - \sigma^2 d| \geq \tau\sigma^2 d) \leq 2m\exp\left(-c\min\left\{\tau^2 d, \tau d\right\}\right).$$

In both cases, we can see that to obtain a μ-incoherent codebook with probability $1-\delta$, is it sufficient to choose:

$$d = O\left(\frac{2}{\mu^2}\ln\frac{m}{\delta}\right)$$

Or, stated another way, we have $\mu = O(\sqrt{(\ln m)/d})$ with high probability. The key point in the two examples above is that the encoding dimension is inversely proportional to μ^2. Per Theorem 1, to decode correctly it is sufficient to have $\mu = 1/(2s)$, meaning that the encoding dimension scales quadratically with the number of elements in the set, but only logarithmically in the alphabet size and probability of error.

We will also consider a third example in which the codewords are *sparse* and binary. However, we defer this for the time being as slightly different encoding methods and analysis techniques are appropriate.

Decoding with Small Probability of Error

The analysis above gives strong *uniform* bounds showing that, with probability at least $1-\delta$ over random choice of the codebook, *every* subset of size at most s will be correctly decoded. However, this guarantee requires us to impose the unappealing restriction that $s \ll \sqrt{d}$ which is a significant practical limitation. We here show that we can obtain $s = O(d)$ but with a weaker *pointwise* guarantee: any arbitrarily chosen set of size at most s will be correctly decoded with probability $1-\delta$ over the random choice of codewords. Rather than insist on a hard upper bound on the incoherence of the codebook, we can instead require the milder condition that random sums over dot-products between $\leq s$ codewords are small with high-probability. We define this property more formally as follows:

Definition 3. ***Subset Incoherence***. *For $\tau > 0$, we say a random mapping $\phi : \mathscr{A} \to \mathscr{H}$ satisfies (s, τ, δ)-subset incoherence if, for any $\mathscr{S} \subset \mathscr{A}$ of size at most s, with probability at least $1-\delta$ over the choice of ϕ:*

$$\max_{a \notin \mathscr{S}} \left| \sum_{a' \in \mathscr{S}} \langle \phi(a), \phi(a') \rangle \right| \leq \tau L^2$$

where $L = \min_{a \in \mathscr{A}} ||\phi(a)||$.

Once again, it turns out that sampling the codewords from a sub-Gaussian distribution can readily be seen to satisfy a subset-incoherence condition with high-probability:

Theorem 4. *Let ϕ be σ-sub-Gaussian and fix some $\mathscr{S} \subset \mathscr{A}$ of size s. Then*

$$\mathbb{P}\left(\max_{a \notin \mathscr{S}} \left| \sum_{a' \in \mathscr{S}} \langle \phi(a), \phi(a') \rangle \right| \geq \tau L^2 \right) \leq 2m \exp\left(-\frac{\kappa \tau^2 L^2}{2s\sigma^2} \right)$$

where κ and L are as in Theorem 3.

Proof. Fix some $a \notin \mathscr{S}$. As described in Theorem 3, the quantity $\langle \phi(a), \phi(a') \rangle$ is sub-Gaussian with parameter at most $L_{\max}^2 \sigma^2$, where $L_{\max} = \max_a \|\phi(a)\|$. Then, again using the fact that sub-Gaussianity is preserved under sums, by Hoeffding's inequality we have:

$$\mathbb{P}\left(\left| \sum_{a' \in \mathscr{S}} \langle \phi(a), \phi(a') \rangle \right| \geq \tau L^2 \right) \leq 2\exp\left(-\frac{\tau^2 L^4}{2sL_{\max}^2 \sigma^2} \right) \leq 2\exp\left(-\frac{\kappa \tau^2 L^2}{2s\sigma^2} \right)$$

where $\kappa = L^2 / L_{\max}^2$. The result follows by union bounding over all m possible a. $\square$

The proof is similar to Theorem 3 and is available in the appendix. As a concrete example, in the practically relevant case that $\phi \sim \{\pm 1\}^d$ the above boils down to:

$$\mathbb{P}\left(\exists a \notin \mathscr{S} \text{ s.t. } \left| \sum_{a' \in \mathscr{S}} \langle \phi(a), \phi(a') \rangle \right| \geq \tau d \right) \leq 2m\exp\left(-\frac{\tau^2 d}{2s} \right).$$

Stated another way, we have: $\tau = O(\sqrt{(s\ln m)/d})$. Following Theorem 1, in order to ensure correct decoding with high probability, we must simply argue that the codebook satisfies the subset-incoherence property with $\tau = 1/2$, meaning we should choose the encoding dimension to be $d = O(s \ln m)$.

This method of analysis is similar to that of [105, 53, 48], who reach the same conclusion vis-à-vis linear scaling using the central limit theorem. However, our formalism is more general and is non-asymptotic.

Comparing Set Representations

We can estimate the size of a set by computing the norm of its encoding, where the precision of the estimate can be bounded in terms of the incoherence of ϕ. In the following discussion, we make the simplifying assumption that the codewords are all of a constant length L. Again appealing to Theorem 2, we can see that this assumption is not onerous since the codeword lengths concentrate around their expected value.

Theorem 5. *Let $\mathscr{S}$ be a set of size s. Then:*

$$s(1-s\mu) \leq \frac{1}{L^2}\|\phi(\mathscr{S})\|_2^2 \leq s(1+s\mu)$$

Proof. The proof is by direct manipulation:

$$\begin{aligned}\frac{1}{L^2}\|\phi(\mathscr{S})\|_2^2 = \frac{1}{L^2}\langle\phi(\mathscr{S}),\phi(\mathscr{S})\rangle &= \frac{1}{L^2}\sum_{a\in\mathscr{S}}\langle\phi(a),\phi(a)\rangle + \frac{1}{L^2}\sum_{a,a'\neq a\in\mathscr{S}}\langle\phi(a),\phi(a')\rangle \\ &\leq \frac{1}{L^2}(sL^2+s^2\mu L^2).\end{aligned}$$

The other direction is analogous. □

Given a pair of sets $\mathscr{S},\mathscr{S}'$ over the same alphabet, we can estimate the size of their intersection and union directly from their encoded representation.

Theorem 6. *Let $\mathscr{S}$ and $\mathscr{S}'$ be sets of size s and s' drawn from $\mathscr{A}$ and denote their encodings by $\phi(\mathscr{S})$ and $\phi(\mathscr{S}')$ respectively.*

$$|\mathscr{S}\cap\mathscr{S}'| - ss'\mu \leq \frac{1}{L^2}\langle\phi(\mathscr{S}),\phi(\mathscr{S}')\rangle \leq |\mathscr{S}\cap\mathscr{S}'| + ss'\mu$$

Proof. Expanding the dot product between the two representations:

$$\begin{aligned}\frac{1}{L^2}\langle\phi(\mathscr{S}),\phi(\mathscr{S}')\rangle &= \frac{1}{L^2}\sum_{a\in\mathscr{S}\cap\mathscr{S}'}\langle\phi(a),\phi(a)\rangle + \frac{1}{L^2}\sum_{a\in\mathscr{S}}\sum_{a'\in\mathscr{S}'\setminus\{a\}}\langle\phi(a),\phi(a')\rangle \\ &\leq |\mathscr{S}\cap\mathscr{S}'| + ss'\mu.\end{aligned}$$

The other direction is analogous. □

Noting as well that $|\mathscr{S}\cup\mathscr{S}'| = |\mathscr{S}| + |\mathscr{S}'| - |\mathscr{S}\cap\mathscr{S}'|$, we see that we can estimate the size of the union using the previous theorem. In practice, it may be unnecessary to compute these quantities with a high degree of precision. For instance, it may only be necessary to identify sets with a large intersection-over-union. Provided the definition of “large” is somewhat loose, we can accept a higher incoherence among the codewords in exchange for reducing the encoding dimension.

Sparse and Low-Precision Encodings

In the previous discussion, we assumed the bundling operator was the element-wise sum. This is a natural choice when the codewords are dense or non-binary. However, the resulting encodings are of unconstrained precision which may be undesirable from a computational perspective. For the purposes of representing sets of size $\leq s$, we may truncate $\phi(\mathscr{S})$ to lie in the range $[-c, c]$, with negligible loss in accuracy provided $c = O(\sqrt{s})$. In practice, it is common to quantize the encodings more aggressively to binary precision by thresholding [73, 115, 18, 63]. In other words, we encode as $\phi(\mathscr{S}) = g_t(\mathscr{S})$, where g_t is a thresholding function that is applied coordinate-wise: $g_t(x) = 1$ if $x \geq t$ and 0 otherwise.

As a notable special case of the thresholding rule described above, we here consider encoding with *sparse* codewords. In this case, we assume that a coordinate in a codeword is non-zero with some small probability. In other words, $\phi(a)_i \sim \text{Bernoulli}(p)$, where $p \ll 1/2$. We may then bundle items by taking an element-wise sum of their codewords with threshold $t = 1$, which is equivalent to taking the element-wise maximum over the codewords. That is, $\phi(\mathscr{S}) = \max_{a \in \mathscr{S}} \phi(a)$, where the max operator is applied coordinate-wise. Noting that the max is upper bounded by the sum in this setting, the notion of incoherence is a relevant quantity and the analysis of Theorem 1 continues to apply.

This encoding procedure is essentially a standard implementation of the popular “Bloom filter” data structure for representing sets [14]. The conventional Bloom filter decoding rule is to threshold $\langle \phi(a), \phi(\mathscr{S}) \rangle$ at $\|\phi(a)\|_1$. There is a large literature on Bloom filters with applications ranging from networking and database systems to neural coding, and several schemes for generating good codewords have been proposed [16, 102, 37]. Using the random coding scheme described here, the optimal value of p can be seen to be $(\ln 2)/s$ and, to ensure the probability of a false positive is at most δ, the encoding dimension should be chosen on the order of $s\ln(1/\delta)$ [16]. A practical benefit of Bloom filters is that they have an efficient implementation using hash functions which does not require materializing a codebook as in methods based on random sampling. This may be beneficial when the alphabet size is large enough that storing codewords is not possible. The connections between Bloom filters and HDC were first noted in [84] who give an interesting extension to the basic Bloom filter that allows it to dynamically resize its capacity.

We remark that this method of encoding is related to an interesting procedure known as "context dependent thinning" (CDT) which can be used to control the density of binary representations [110, 85]. CDT takes the logical "and" of $\phi(\mathscr{S})$ and some permutation $\sigma(\phi(\mathscr{S}))$ to obtain the thinned representation $\phi(\mathscr{S})' = \phi(\mathscr{S}) \wedge \sigma(\phi(\mathscr{S}))$. This process can be repeated until the desired density of $\phi(\mathscr{S})$ is achieved. A capacity analysis of CDT representations can be found in [85].

2.1.2 Robustness to Noise

In this section we explore the noise robustness properties of the encoding methods discussed above using the formalism of incoherence. We consider some unspecified noise process which corrupts the encoding of a set $\mathscr{S} \subset \mathscr{A}$ of size at most s according to $\tilde{\phi}(\mathscr{S}) = \phi(\mathscr{S}) + \Delta_{\mathscr{S}}$. We say $\Delta_{\mathscr{S}}$ is ρ-bounded if:

$$\max_{a\in\mathscr{A}} |\langle \phi(a), \Delta_{\mathscr{S}} \rangle| \leq \rho.$$

We are interested in understanding the conditions under which we can still decode reliably.

Theorem 7. *Suppose $\mathscr{S}$ has size $\leq s$ and $\Delta_{\mathscr{S}}$ is ρ-bounded. We can correctly decode $\mathscr{S}$ using the thresholding rule from Theorem 1 if:*

$$\frac{\rho}{L^2} + s\mu < \frac{1}{2}$$

where $L = \min_{a\in\mathscr{A}} \|\phi(a)\|_2$.

Proof. Consider some symbol $a \in \mathscr{A}$. In the event $a \in \mathscr{S}$:

$$\langle \phi(a), \phi(\mathscr{S}) + \Delta_{\mathscr{S}} \rangle = \langle \phi(a), \phi(\mathscr{S}) \rangle + \langle \phi(a), \Delta_{\mathscr{S}} \rangle \geq L^2 - sL^2\mu - \rho$$

and when $a \notin \mathscr{S}$:

$$\langle \phi(a), \phi(\mathscr{S}) + \Delta_{\mathscr{S}} \rangle \leq sL^2\mu + \rho$$

Therefore we can decode correctly if:

$$\frac{\rho}{L^2} + s\mu < \frac{1}{2}$$

$\square$

The practical implication is that there is a tradeoff between the incoherence of the codebook and robustness to noise: a higher incoherence allows for a smaller encoding dimension but at the cost of a tighter constraint on ρ. We can analyze several practically relevant noise models by placing additional restrictions on $\Delta_{\mathscr{S}}$ and by considering worst or typical case bounds on ρ. We here consider different forms of noise under constraints on $\mathscr{H}$. Our goal is to understand how the magnitude of noise that can be tolerated scales with the encoding dimension, size s of the encoded set, and size m of the alphabet. In each setting we consider a "passive" model in which the noise is sampled randomly from some distribution, and an "adversarial" model in which the noise is arbitrary and may be designed to maliciously corrupt the encodings. We again appeal to Theorem 2 to justify a simplifying assumption that the codewords are of equal length.

Lemma 8. ***Sub-Gaussian Codewords***. *Fix a centered and σ-sub-Gaussian codebook ϕ whose codewords are of length L. Consider the passive additive white Gaussian noise model $\Delta_{\mathscr{S}} \sim \mathscr{N}(0, \sigma_\Delta^2 I_d)$; that is, each coordinate is corrupted by Gaussian noise with mean zero and variance σ_Δ^2. Then, we can correctly decode with probability $1 - \delta$ over random draws of $\Delta_{\mathscr{S}}$*

provided:

$$\sigma_\Delta < \frac{L}{\sqrt{2\ln(2m/\delta)}}\left(\frac{1}{2} - s\mu\right)$$

Now consider an adversarial model in which $\Delta_{\mathscr{S}}$ is arbitrary save for a constraint on the norm: $\|\Delta_{\mathscr{S}}\|_2 \leq \omega L$. Then, we can correctly decode provided:

$$\omega < \frac{1}{2} - s\mu$$

Proof. Let us first consider the passive case in which $\Delta_{\mathscr{S}} \sim \mathscr{N}(0, \sigma_\Delta^2 I_d)$. Fix some $a \in \mathscr{A}$. Then $\langle \phi(a), \Delta_{\mathscr{S}} \rangle \sim \mathscr{N}(0, \sigma_\Delta^2 L^2)$. By a standard tail bound on the Gaussian distribution [151] and the union bound, we have:

$$\mathbb{P}(\exists a \text{ s.t. } |\langle \phi(a), \Delta_{\mathscr{S}} \rangle| \geq \rho) \leq 2m \exp\left(-\frac{\rho^2}{2\sigma_\Delta^2 L^2}\right).$$

Therefore, with probability $1-\delta$, we have that $\Delta_{\mathscr{S}}$ is ρ-bounded for

$$\rho \leq \sigma_\Delta L \sqrt{2\ln(2m/\delta)}.$$

By Theorem 7 we can decode correctly if:

$$\frac{\sigma_\Delta L \sqrt{2\ln(2m/\delta)}}{L^2} + s\mu < \frac{1}{2} \Rightarrow \sigma_\Delta < \frac{L}{\sqrt{2\ln(2m/\delta)}}\left(\frac{1}{2} - s\mu\right).$$

Now consider the adversarial case in which $\|\Delta_{\mathscr{S}}\|_2 \leq \omega L$. By the Cauchy-Schwarz inequality, $|\langle \phi(a), \Delta_{\mathscr{S}} \rangle| \leq \omega L^2$. Therefore, by Theorem 7, we can decode correctly if

$$\frac{\omega L^2}{L^2} + s\mu < \frac{1}{2} \Rightarrow \omega < \frac{1}{2} - s\mu.$$

$\square$

We again emphasize that, per Theorem 3, $\mu = O(\sqrt{(\ln m)/d})$. Since $L = O(\sqrt{d})$, we can see that we can tolerate $\sigma_\Delta \approx \sqrt{d/(\ln m)} - s$ in the passive case. We next turn to a notable special case of the above in which the codewords are dense and binary. In this case, we may assume that $\mathscr{H}$ is constrained to be integers in the range $[-s, s]$.

Lemma 9. ***Dense Binary Codewords***. *Fix a codebook ϕ such that $\phi(a) \sim \{\pm 1\}^d$ for each $a \in \mathscr{A}$. Consider a passive noise model in which $\Delta_{\mathscr{S}} \sim \text{unif}(\{-c, ..., c\}^d)$; that is, each coordinate is shifted by an integer amount chosen uniformly at random between $-c$ and c. Then, we can correctly decode with probability $1-\delta$ provided:*

$$c < \sqrt{\frac{d}{2\ln(2m/\delta)}} \left(\frac{1}{2} - s\mu\right)$$

Now consider an adversarial model in which we assume $\|\Delta_{\mathscr{S}}\|_1 \leq \omega s d$. Then we can decode correctly if:

$$\omega < \frac{1}{2s} - \mu.$$

Proof. Consider first the case of passive noise. Fix some $a \in \mathscr{A}$. Noting that $\langle \phi(a), \Delta_{\mathscr{S}} \rangle$ is the sum of d terms bounded in $[-c, c]$, another application of Hoeffding's inequality and the union bound will show:

$$\mathbb{P}(\exists a \text{ s.t. } |\langle \phi(a), \Delta_{\mathscr{S}} \rangle| \geq \rho) \leq 2m \exp\left(-\frac{\rho^2}{2c^2 d}\right).$$

Therefore, with probability $1-\delta$, we have that $\Delta_{\mathscr{S}}$ is ρ-bounded for $\rho \leq c\sqrt{2d\ln(2m/\delta)}$. Noting that $L = \sqrt{d}$ exactly, the result follows by applying Theorem 7.

Now let us consider the adversarial case in which $\|\Delta_{\mathscr{S}}\|_1 \leq \omega s d$. We first observe that $|\langle \phi(a), \Delta_{\mathscr{S}} \rangle| \leq \|\phi(a)\|_\infty \|\Delta_{\mathscr{S}}\|_1 \leq \omega s d$. Then, applying Theorem 7 we obtain:

$$\frac{\omega s d}{d} + s\mu < \frac{1}{2} \Rightarrow \omega < \frac{1}{2s} - \mu$$

as claimed. $\square$

We next consider the case of Section 2.1.1 in which the codewords are sparse and binary and the bundling operator is the element-wise maximum. We here assume that $\tilde{\phi}(\mathscr{S}) = \phi(\mathscr{S}) + \Delta_{\mathscr{S}}$ is truncated so that each coordinate is either 0 or $+1$.

Lemma 10. ***Sparse Binary Codewords***. *Fix a codebook ϕ such that $\phi(a) \in \{0,1\}^d$, and assume some fraction $p \ll 1/2$ of coordinates are non-zero for each $a \in \mathscr{A}$. Consider a passive noise model in which:*

$$\Delta_{\mathscr{S}} \sim \begin{cases} -1 & w.p.\ \frac{\theta}{2} \\ 0 & w.p.\ 1-\theta \\ +1 & w.p.\ \frac{\theta}{2}. \end{cases}$$

Then we can decode correctly with probability $1-\delta$ provided:

$$\theta < \frac{1}{2} - 2s\mu - \sqrt{\frac{1}{2dp}\ln\frac{2m}{\delta}}.$$

Now consider an adversarial model in which $\|\Delta_{\mathscr{S}}\|_1 \leq \omega d$. Then we can decode correctly if $\omega < p(\frac{1}{2} - s\mu)$.

Proof. Consider first the passive noise model. Fix some $\phi(a)$. Then:

$$|\langle \phi(a), \Delta_{\mathscr{S}} \rangle| \leq \sum_{i=1}^{d} |\phi(a)^{(i)} \Delta_{\mathscr{S}}^{(i)}|.$$

Treating $\phi(a)$ as a fixed vector with dp non-zero entries, the sum is concentrated in the range $dp(\theta \pm \varepsilon)$, and so $\rho \leq dp(\theta + \varepsilon)$ with high probability. By Chernoff/Hoeffding and the union-bound, with probability $1-\delta$:

$$\varepsilon \leq \sqrt{\frac{1}{2dp}\ln\frac{2m}{\delta}}.$$

The result is obtained by noting that $L = \sqrt{pd}$ and applying Theorem 7.

For the adversarial case, the result is obtained by again observing that $|\langle \phi(a), \Delta_{\mathscr{S}} \rangle| \leq \|\phi(a)\|_\infty \|\Delta_{\mathscr{S}}\|_1 \leq \omega d$ for any $a \in \mathscr{A}$ and applying Theorem 7. □

2.2 Encoding Structures

We are often interested in representing more complex data types, such as objects with multiple attributes or "features." In general, we suppose that we observe a set of features $\mathscr{F}$ whose values are assumed to lie in some set $\mathscr{A}$. Let $\psi : \mathscr{F} \rightarrow \mathscr{H}$ be an embedding of features, and $\phi : \mathscr{A} \rightarrow \mathscr{H}$ be an embedding of values. We associate a feature with its value through use of the *binding* operator $\otimes : \mathscr{H} \times \mathscr{H} \rightarrow \mathscr{H}$ that creates an embedding for a (feature,value) pair. For a feature $f \in \mathscr{F}$ taking on a value $a \in \mathscr{A}$, its embedding is constructed as $\psi(f) \otimes \phi(a)$. A data point $x = \{(f_i \in \mathscr{F}, x_i \in \mathscr{A})\}_{i=1}^{n}$ consists of n such pairs. For simplicity, we assume each x possesses all attributes, although our analysis also applies to the case that x possesses only some subset of attributes. The entire embedding for x is constructed as [105]:

$$\phi(x) = \bigoplus_{i=1}^{n} \psi(f_i) \otimes \phi(x_i) \tag{2.1}$$

As with sets we would typically like $\phi(x)$ to be *decodable* in the sense that we can recover the value associated with a particular feature, and *comparable* in the sense that $\langle \phi(x), \phi(x') \rangle$ is reflective of a reasonable notion of similarity between x and x'.

From a formal perspective, we require the binding operator to satisfy several properties. First, binding should be associative and commutative. That is, for all $a, b, c \in \mathscr{H}$, $(a \otimes b) \otimes c = a \otimes (b \otimes c)$, and $a \otimes b = b \otimes a$. Second, there should exist an identity element $I \in \mathscr{H}$ such that $I \otimes a = a$ for all $a \in \mathscr{H}$. Third, for all $a \in \mathscr{H}$, there should exist some $a^{-1} \in \mathscr{H}$ such that $a \otimes a^{-1} = I$. These properties are equivalent to stipulating that $\mathscr{H}$ be an abelian group under $\otimes$. Furthermore, binding should distribute over bundling. That is, for any $a, b, c \in \mathscr{H}$, it should be the case that $a \otimes (b + c) = a \otimes b + a \otimes c$. We here also require that the lengths of bound pairs are bounded, that is to say: $\max_{f \in \mathscr{F}, a \in \mathscr{A}} \|\psi(f) \otimes \phi(a)\|_2 \leq M$.

A natural choice of embedding satisfying these properties is to sample $\psi(f)$ randomly from $\{\pm 1\}^d$ and choose $\otimes$ to be the element-wise product. In this case $\psi(f)$ is its own inverse, that is $\psi(f) \otimes \psi(f) = I$, and binding preserves lengths of codewords. We focus on this case here as it is intuitive, but our analysis generalizes in a straightforward way to any particular implementation satisfying the properties listed above. One can see the bound pairs satisfy various incoherence properties with high probability. For instance, we may declare the binding to be μ-incoherent if:

$$\max_{a \in \mathscr{A}} \max_{a' \in \mathscr{A}, f \in \mathscr{F}} \langle \phi(a), \psi(f) \otimes \phi(a') \rangle \leq \mu L^2$$

where $L = \min_{a \in \mathscr{A}} \|\phi(a)\|_2$. We can extend Theorem 3 to see this property is satisfied with high probability:

Theorem 11. *Fix $d, n, m \in \mathbb{Z}^+$ and $\mu \in \mathbb{R}^+$. Let ϕ be centered and σ-sub-Gaussian, $\otimes$ be the element-wise product, and $\psi(f) \sim \{\pm 1\}^d$. Then:*

$$\mathbb{P}(\exists a, a' \in \mathscr{A}, f \in \mathscr{F} \text{ s.t. } |\langle \phi(a), \phi(a') \otimes \psi(f) \rangle| \geq \mu L^2) \leq nm^2 \exp\left(-\frac{\kappa \mu^2 L^2}{2\sigma^2}\right)$$

where $L = \min_{a \in \mathscr{A}} \|\phi(a)\|_2$ and κ is as defined in Theorem 3.

Proof. Note first that $\|\phi(a) \otimes \psi(f)\|_2 = \|\phi(a)\|_2$. Then, fixing a, a' and f, by Hoeffding's inequality:

$$\mathbb{P}(|\langle \phi(a), \phi(a') \otimes \psi(f) \rangle| \geq \mu L^2) \leq 2 \exp\left(-\frac{L^4 \mu^2}{2\sigma^2 ||\phi(a')||_2^2}\right) \leq 2 \exp\left(-\frac{k \mu^2 L^2}{2\sigma^2}\right)$$

where we have again defined $\kappa = (\min_a \|\phi(a)\|_2^2)/(\max_{a'} \|\phi(a')\|_2^2)$. The result follows by the union bound over all $< nm^2/2$ combinations of a, a', f. □

This result is appealing because it means that the incoherence scales only logarithmically with $m \times n$ which may be large in practice. As a corollary to the previous theorem, we also obtain the following useful incoherence property:

$$\mathbb{P}(\exists a, a', f \neq f' \text{ s.t. } |\langle \phi(a), (\phi(a') \otimes \psi(f)) \otimes \psi^{-1}(f') \rangle| \geq \mu L^2) \leq m^2 n^2 \exp\left(-\frac{\kappa \mu^2 L^2}{2\sigma^2}\right) \tag{2.2}$$

where $\psi^{-1}(f)$ is the inverse of $\psi(f)$ with respect to $\otimes$. This notion of incoherence is useful for decoding representations. Along similar lines:

$$\mathbb{P}(\exists a, a', f \neq f' \text{ s.t. } |\langle \phi(a) \otimes \psi(f), \phi(a') \otimes \psi(f') \rangle| \geq \mu L^2) \leq m^2 n^2 \exp\left(-\frac{\kappa \mu^2 L^2}{2\sigma^2}\right) \tag{2.3}$$

We note that the previous statement refers to symbols associated with different attributes and thus does not require any particular incoherence assumption on the $\phi(a)$.

2.2.1 Decoding Structures

This representation can be decoded to recover the value associated with a particular feature. To recover the value of the *i*-th feature, we use the following rule:

$$\hat{x}_i = \underset{a \in \mathscr{A}}{\operatorname{argmax}} \ \langle \phi(a), \phi(x) \otimes \psi^{-1}(f_i) \rangle$$

where $\psi^{-1}(f)$ denotes the group inverse of $\psi(f)$. Since the binding operator is assumed to distribute over bundling, the dot-product above expands to:

$$\langle \phi(a), \phi(x_i) \rangle + \sum_{j \neq i} \langle \phi(a), (\phi(x_j) \otimes \psi(f_j)) \otimes \psi^{-1}(f_i) \rangle$$
$$\begin{cases} \geq L^2(1 - n\mu) & \text{if } x_i = a \\ \leq nL^2\mu & \text{otherwise} \end{cases}$$

where the incoherence can be bounded as as in Equation 2.2. Thus $\mu < 1/(2n)$ is a sufficient condition for decodability.

2.2.2 Comparing Structures

As with sets, we may wish to compare two structures without decoding them. As one would expect given Theorem 6, this is can be achieved by computing the dot-product between their encodings:

Theorem 12. *Let x and x' be two structures drawn from a common alphabet $\mathscr{F} \times \mathscr{A}$. Denote their encodings using Equation 2.1 by $\phi(x)$ and $\phi(x')$. Then, if binding is μ-incoherent:*

$$|x \cap x'| - n^2\mu \leq \frac{1}{L^2}\langle \phi(x), \phi(x') \rangle \leq |x \cap x'| + n^2\mu$$

where $x \cap x'$ is defined to be the set $\{i : x_i = x'_i\}_{i=1}^{n}$, that is, the features on which x and x' agree.

Proof. Expanding:

$$\langle \phi(x), \phi(x') \rangle = \langle \sum_{i=1}^{n} \phi(x_i) \otimes \psi(f_i), \sum_{j=1}^{n} \phi(x'_j) \otimes \psi(f_j) \rangle$$

$$= \sum_{i=1}^{n} \langle \phi(x_i) \otimes \psi(f_i), \phi(x'_i) \otimes \psi(f_i) \rangle + \sum_{i \neq j} \langle \phi(x_i) \otimes \psi(f_i), \phi(x'_j) \otimes \psi(f_j) \rangle$$

A term in the first sum is L^2 if $x_i = x'_i$ and bounded in $\pm L^2\mu$ otherwise. So the expression above is bounded as:

$$\leq L^2|x \cap x'| + L^2 n^2 \mu$$

and the other direction of the inequality is analogous. □

As a practical example, in bioinformatics it is common to search for regions of high similarity between a "reference" and "query" genome. Work in [66] and [78] explored the use HD computing to accelerate this process by encoding short segments of DNA and estimating similarity on the HD representations.

2.2.3 Encoding Sequences

Sequences are an important form of structured data. In this case, the feature set is simply the list of positions $\{1,2,3,...\}$ in the sequence. In practical applications, we are often interested in streams of data which arrive continuously over time. Typically, real-world processes do not exhibit infinite memory and we only need to store the $n \geq 1$ most recent observations at any time. In the streaming setting, we would like to avoid needing to fully re-encode all n data points each time we receive a new sample, as would be the case using the method described above. This motivates the use of shift based encoding schemes [75, 118, 79]. Let $\rho^{(i)}(z)$ denote a cyclic left-shift of the elements of z by i coordinates, and $\rho^{(-i)}(z)$ denote a cyclic right-shift by i coordinates. In other words: $\rho^{(1)}((z_1,z_2,\ldots,z_{d-1},z_d)) = (z_2,z_3,\ldots,z_d,z_1)$. In shift-based encoding a sequence $x = (x_1,...,x_n)$ is represented as:

$$\phi(x) = \bigoplus_{i=1}^{n} \rho^{(n-i)}(\phi(x_i)),$$

where we take $\oplus$ to be the element wise sum. Now suppose we receive symbol $n+1$ and wish to append it to $\phi(x)$ while removing $\phi(x_1)$. Then we may apply the rule:

$$\rho^{(1)}(\phi(x) - \rho^{(n-1)}(\phi(x_1))) \oplus \phi(x_{n+1}) = \bigoplus_{i=1}^{n} \rho^{(n-i)}\phi(x_{i+1})$$

where we can additionally note that ρ is a special type of permutation and that permutations distribute over sums. However, in order to decode correctly, each $\phi(a)$ must satisfy an incoherence condition with the $\rho^{(j)}(\phi(a'))$. We can again use the randomly generated nature of the codewords to argue this is the case; however, we must here impose the additional restriction that the $\phi(a)$ be bounded, and accordingly restrict attention to the case $\phi(a) \sim \{\pm 1\}^d$.

Theorem 13. *Fix $d,m,n < d \in \mathbb{Z}^+$ and $\mu \in \mathbb{R}^+$ and let $\phi(a) \sim \{\pm 1\}^d$. Then:*

$$\mathbb{P}(\exists a,a' \in \mathscr{A}, i \neq 0 \text{ s.t. } |\langle \phi(a), \rho^{(i)}(\phi(a'))\rangle| \geq \mu d) \leq nm^2 \exp\left(-\frac{\mu^2 d}{4}\right)$$

Proof. Fix some a, a' and i. In the case that $a \neq a'$, $\phi(a)$ and $\rho^{(i)}(\phi(a))$ are mutually independent. However, when $a = a'$, $\phi(a)$ and $\rho^{(i)}(\phi(a))$ only satisfy pairwise independence and the techniques of Theorem 3 cannot be applied. To resolve this difficulty, let $f(\phi(a)) = \langle \phi(a), \rho^{(i)}(\phi(a)) \rangle$, and denote by $\phi(a)^{\setminus k}$ the vector formed by replacing the k-th coordinate in $\phi(a)$ with an arbitrary value $\in \{+1, -1\}$. Then $|f(\phi(a)) - f(\phi(a)^{\setminus k})| \leq 4$ and so by the bounded-differences inequality [92]:

$$\mathbb{P}(|\langle \phi(a), \rho^{(i)}(\phi(a')) \rangle| \geq \mu d) \leq 2\exp\left(-\frac{\mu^2 d}{4}\right).$$

The result follows by the union bound. □

Several other related methods for encoding sequential information have been proposed in the literature [105, 53]. For an extensive discussion of these approaches as well as an interesting discussion involving sequences of infinite length, the reader is referred to [48].

2.2.4 Discussion and Comparison with Prior Work

We conclude our treatment of encoding and decoding discrete data with some brief discussion of our approach and its relation to antecedents in the literature. A key question addressed here and by several pieces of prior work is to bound the magnitude of crosstalk noise in terms of the encoding dimension (d), the number of items to encode (s) and the alphabet size (m). Early analysis in [105, 53, 85] recovers the same asymptotic relationship as we do, but only under specific assumptions about the method used to generate the codewords and particular instantiations of the bundling and binding operators.

Work in [48] provides a significantly more general treatment which, like ours, aims to abstract away from the particular choice of distribution from which codewords are sampled and from the particular implementation of bundling and binding operator. Their approach assumes the codewords are generated by sampling each component i.i.d. from some distribution and uses the central limit theorem (CLT) to justify modeling the crosstalk noise by a Gaussian distribution.

Error bounds in the non-asymptotic setting are then obtained by applying a Chernoff style bound to the resulting Gaussian distribution. This approach again recovers the same asymptotic relationship between d, s and m as us, but does not generally yield formal bounds in the non-asymptotic setting. Our approach based on sub-Gaussianity formalizes this analysis in the non-asymptotic setting. Like us, [48] also considers the effect of noise on the HD representations, but their treatment is limited to additive white noise, whereas we address both arbitrary additive passive noise and adversarial noise.

In summary, our formalism using the notion of incoherence allows us to decouple the analysis of decoding and noise-robustness from any particular method for generating codewords and readily yields rigorous bounds in the non-asymptotic setting. Our approach is applicable to a large swath of HD computing and enables us to offer more general conditions under which thresholding based decoding schemes will succeed and of the effect of noise than is available in prior work.

2.3 Encoding Euclidean Data

One option for encoding Euclidean vectors is to treat them as a special case of the "structured data" considered in the preceding section. As before, we think of our data as a collection of (feature,value) pairs $x = \{(f_i, x_i)\}_{i=1}^n$ with the important caveat that $x_i \in \mathbb{R}^n$. This case is more complex because the feature values may now be continuous, and because the data possesses geometric structure which is typically relevant for downstream tasks and must be preserved by encoding. We here analyze two of the most widely used methods for encoding Euclidean data and discuss general properties of structure preserving embeddings in the context of HD computing.

2.3.1 Position-ID Encoding

A widely-used method in practice is to quantize the raw signal to a suitably low precision and then apply the structure encoding method discussed in the previous section [111, 112, 85, 116].

In this approach, we first quantize the support of each feature $f \in \mathscr{F}$ into some set of m bins with centroids $a_1 < \cdots < a_m$ and assign each bin a codeword $\phi(a) \in \mathscr{H}$. However, instead of requiring the codewords to be incoherent, we now require the correlation between codewords to reflect the distance between corresponding quantizer bins. In other words $\langle \phi(a), \phi(a') \rangle$ should be monotonically decreasing in $|a - a'|$.

A simple method can be used to generate monotonic codebooks when the codewords are randomly sampled from $\{\pm 1\}^d$ [111, 152]. Fixing some feature f, the codeword for the minimal quantizer bin, $\phi(a_1)$, is generated by sampling randomly from $\{\pm 1\}^d$. To generate the codeword for the second bin, we simply flip some set of $\lceil b \rceil$ bits in $\phi(a_1)$, where:

$$b = \frac{a_2 - a_1}{a_m - a_1} \times \frac{d}{2}$$

The codeword for the third bin is generated analogously from the second, where we assume the bits to be flipped are sampled such that a bit is flipped at most once. Thus the codewords for the minimal and maximal bins are orthogonal and the correlation between codewords for intermediate bins is monotonically decreasing in the distance between their corresponding bin centroids.

In practice, it seems to be typical to use a single codebook for all features and for the quantizer to be a set of evenly spaced bins over the support of the data. While simple, this approach is likely to have sub-optimal rate when the features are on different scales or are far from the uniform distribution. Encoding then proceeds as follows:

$$\phi(x) = \sum_{i=1}^{n} \phi(x_i) \otimes \psi(f_i)$$

where, as before $\psi \in \{\pm 1\}^d$ is a vector which encodes the index i of a feature value x_i as in the previous section on encoding sequences; hence the name "position-ID" encoding. There are several variations on this theme which are compared empirically in [85].

This general encoding method was analyzed by [112], in the specific case of sparse and binary codewords, who show it preserves the L1 distance between points in expectation but do not provide distortion bounds. We here provide such bounds using our formalism of matrix incoherence. We assume that the underlying quantization of the points is sufficiently fine that it is a low-order term that can be ignored.

Theorem 14. *Let x and x' be points in $[0,1]^n$ with encodings $\phi(x)$ and $\phi(x')$ generated using the rule described above. Assume that ϕ satisfies $\langle \phi(a), \phi(a') \rangle = d(1-|a-a'|)$ for all $a, a' \in \mathscr{A}$, and let $\psi \sim \{\pm 1\}^d$. Then, for all x, x':*

$$2d(\|x-x'\|_1 - 2n^2\mu) \leq ||\phi(x)-\phi(x')||_2^2 \leq 2d(\|x-x'\|_1 + 2n^2\mu)$$

The proof is similar to Theorem 12 and is available in the Appendix.

The practical implication of the previous theorem is that the position-ID encoding method preserves the L1 distance between points up to an additive distortion which can be bounded by the incoherence of the codebook. Per Equation 2.3, $\mu = O(\sqrt{\ln(mn)/d})$. Therefore, to ensure that $\frac{1}{d}\|\phi(x)-\phi(x')\|_2^2 \approx \|x-x'\|_1 \pm \varepsilon$, the previous result implies we should choose $d = O(\frac{n^4}{\varepsilon^2}\ln(nm))$. This can be relaxed to a quadratic dependence on n in exchange for a weaker pointwise bound, but in either case means the encoding method may be problematic when the dimension of the underlying data is high.

Noting that $||\phi(x)||_2^2 \in nd \pm n^2 d\mu$, we can see that the encodings of each point are roughly of equal norm and lie in a ball of radius at most $n\sqrt{d\mu}$, where the exact position depends on the instantiation of the codebook. Thus, we can loosely interpret the encoding procedure as mapping the data into a thin shell around the surface of a high dimensional sphere.

2.3.2 Random Projection Encoding

Another popular family of encoding methods embeds the data into $\mathscr{H}$ under some random linear map followed by a quantization [108, 65]. More formally, for some $x \in \mathbb{R}^n$, these embeddings take the form:

$$\phi(s) = g(Mx)$$

where $M \in \mathbb{R}^{d \times n}$ is a matrix whose rows are sampled uniformly at random from the surface of the n-dimensional unit sphere, and g is some optional non-linearity (for example the `sign` function). The embedding matrix M may also be quantized to lower precision. This encoding method has also been studied in the context of kernel approximation where it is used to approximate the angular kernel [28], and to construct low-distortion binary embeddings [69, 104]. While the following result is well known, we here show this encoding method preserves angular distance up to an additive distortion as this fact is important for subsequent analysis.

Theorem 15. *Let $\mathscr{S}^{n-1} \subset \mathbb{R}^n$ denote the n-dimensional unit sphere. Let $M \in \mathbb{R}^{d \times n}$ be a matrix whose rows are sampled uniformly at random from $\mathscr{S}^{n-1}$. Let $\mathscr{X}$ be a set of points supported on $\mathscr{S}^{n-1}$. Denote the embedding of a point by $\phi(x) = \operatorname{sign}(Mx)$. Then, for any $x, x' \in \mathscr{X}$, with high probability:*

$$d\theta - O(\sqrt{d}) \leq d_{ham}(\phi(x), \phi(x')) \leq d\theta + O(\sqrt{d})$$

where $d_{ham}(a,b)$ is the Hamming distance between a and b, defined to be the number of coordinates on which a and b differ, and $\theta = \frac{1}{\pi}\cos^{-1}(\langle x, x' \rangle) \in [0,1]$ is proportional to the angle between x and x'.

Proof. Let $M^{(i)}$ denote the ith row of the matrix M. Then, the ith coordinate in the embedding of x can be written as $\operatorname{sign}(\langle M^{(i)}, x \rangle)$. The probability that the embeddings differ on their ith coordinate, that is $(\langle M^{(i)}, x \rangle)(\langle M^{(i)}, x' \rangle) < 0$, is exactly $\angle(x, x')/\pi$: the angle (in radians) between x and x' divided by π.

Therefore, the number of coordinates on which $\phi(x)$ and $\phi(x')$ disagree is, concentrated in the range, $d(\theta \pm \varepsilon)$. By Chernoff/Hoeffding, we have that with probability $1-\delta$:

$$d\varepsilon \leq \sqrt{2d \ln \frac{2}{\delta}}.$$

$\square$

Noting that $\langle \phi(x), \phi(x') \rangle = d - 2d_{ham}(\phi(x), \phi(x'))$, we obtain the following simple corollary:

Corollary 16. *Let ϕ and θ be as defined in Theorem 15. Then, with high probability:*

$$d(1-2\theta) - O(\sqrt{d}) \leq \langle \phi(x), \phi(x') \rangle \leq d(1-2\theta) + O(\sqrt{d})$$

To obtain a more explicit relationship with the dot product, we can use the first-order approximation $\cos^{-1}(x) \approx (\pi/2) - x$, to obtain $\theta \approx \frac{1}{2} - \frac{1}{\pi}\langle x, x' \rangle$, from which we obtain:

$$d(1-2\theta) \approx \frac{2d}{\pi}\langle x, x' \rangle.$$

We emphasize that, in comparison to the position-ID method, the distortion in this case does not depend on the dimension of the underlying data which means this method may be preferable when the data dimension is large.

Connection with Kernel Approximation

A natural question is whether the encoding procedure described above, which preserves angles, can be generalized to capture more diverse notions of similarity? We can answer in the affirmative by noting that the random projection encoding method is closely related to the notion of random Fourier features which have been widely used for kernel approximation [119]. The basic idea is to construct an embedding $\phi : \mathbb{R}^n \to \mathbb{R}^d$, such that $\langle \phi(x), \phi(x') \rangle \approx k(x, x')$, where k is a shift-invariant kernel. The construction exploits the fact that the Fourier transform of a shift-invariant kernel k is a probability measure: a well known result from harmonic analysis known as Bochner's Theorem [124]. The embedding itself is given by $\phi(x) = \frac{1}{\sqrt{d}}\cos(Mx+b)$, where the rows of M are sampled from the distribution induced by k and the coordinates of b are sampled uniformly at random from $[0,2\pi]$.

Subsequent work in [113] gave a simple scheme for quantizing the embeddings produced from random Fourier features to binary precision. Their construction yields an embedding $\psi : \mathbb{R}^n \to \{0,1\}^d$ such that:

$$f_1(k(x,x')) - \Delta \leq \frac{1}{d} d_{ham}(\psi(x), \psi(x')) \leq f_2(k(x,x')) + \Delta$$

where $f_1, f_2 : \mathbb{R} \to \mathbb{R}$ are independent of the choice of kernel, and Δ is a distortion term. The embedding itself is constructed by applying a quantizer $Q_t(x) = \text{sign}(x+t)$ coordinate wise over the embeddings constructed from random Fourier features. In other words $\psi(x)_i = \frac{1}{2}(1 + Q_{t_i}(\phi(x)_i))$, where $t_i \sim \text{Unif}[-1,1]$, and $\phi(x)$ is a random Fourier feature.

This connection is highly appealing for HD computing. The quantized random Fourier feature scheme presents a simple recipe for constructing encoding methods meeting the desiderata of HD computing while preserving a rich variety of structure in data. For instance, shift-invariant kernels preserving the L1 and L2 distance—among many others—can be approximated using the method discussed above. Furthermore, this observation provides a natural point of contact between HD computing and the vast literature on kernel methods which has produced a wealth of algorithmic and theoretical insights.

2.3.3 Consequences of Distance Preservation

The encoding methods discussed above are both appealing because they preserve reasonable notions of distance between points in the original data. Distance preservation is a sufficient condition to establish other desirable properties of encodings, namely preservation of neighborhood/cluster structure, robustness to various forms of noise, and in some cases, preservation of linear separability. We address the first two items here and defer the latter for our discussion of learning on HD representations. We formalize our notion of distance preservation as follows:

Definition 4. ***Distance-Preserving Embedding****: Let $\delta_{\mathscr{X}}$ be a distance function on $\mathscr{X} \subset \mathbb{R}^n$ and δ_H be a distance function on $\mathscr{H}$. We say ϕ preserves $\delta_{\mathscr{X}}$ under δ_H if, there exist functions $\alpha, \beta : \mathbb{Z}^+ \to \mathbb{R}$ such that $\beta(d)/\alpha(d) \to 0$ as $d \to \infty$, and:*

$$\alpha(d)\delta_{\mathscr{X}}(x,x') - \beta(d) \leq \delta_{\mathscr{H}}(\phi(x),\phi(x')) \leq \alpha(d)\delta_{\mathscr{X}}(x,x') + \beta(d) \tag{2.4}$$

for all $x, x' \in \mathscr{X}$.

We typically wish the distance function $\delta_{\mathscr{H}}$ on $\mathscr{H}$ to be simple to compute. In practice, it is often taken to be the Euclidean, Hamming, or angular distance. The position-ID method preserves the L1 distance with δ_H the squared Euclidean distance, $\alpha(d) = 2d$, and $\beta(d) \leq n^2\mu d$; recall that in the constructions above, μ scales inversely with d and thus $\beta(d)/\alpha(d) \to 0$. The signed random-projection method preserves the angular distance with $\alpha(d) = O(d)$, $\beta(d) = O(\sqrt{d})$, and δ_H the Hamming, angular, or Euclidean distance.

Preservation of Cluster Structure

In general, there is no universally applicable definition of cluster structure. Indeed, numerous algorithms have been proposed in the literature to target various reasonable notions of what constitutes a "cluster" in the data. Preservation of a distance function accords naturally with K-means like algorithms which, given a set of data $\mathscr{X} \subset \mathbb{R}^n$ compute a set of centroids $\mathscr{C} = \{c_i\}_{i=1}^k$, and define associated clusters as the Voronoi cells associated with each centroid. We here adopt this notion and state that cluster structure $\mathscr{C}$ is preserved if, for any $x \in \mathscr{X}$:

$$\operatorname*{argmin}_{c \in \mathscr{C}} \delta_{\mathscr{X}}(x,c) = \operatorname*{argmin}_{c \in \mathscr{C}} \delta_{\mathscr{H}}(\phi(x),\phi(c))$$

In other words, that the set of points bound to a particular cluster centroid does not change under the encoding. We can restate the above as requiring that, for some point x bound to a cluster centroid c, it is the case that:

$$\delta_{\mathscr{H}}(\phi(x),\phi(c)) < \delta_{\mathscr{H}}(\phi(x),\phi(c'))$$

for any $c' \in \mathscr{C} \setminus \{c\}$. From Definition 4 we have:

$$\delta_{\mathscr{H}}(\phi(x),\phi(c')) - \delta_{\mathscr{H}}(\phi(x),\phi(c)) \geq \alpha(d)(\delta_{\mathscr{X}}(x,c') - \delta_{\mathscr{X}}(x,c)) - 2\beta(d)$$

for any $x \in \mathscr{X}$ and $c, c' \in \mathscr{C}$. Rearranging the expressions above we can see the desired property will be satisfied if:

$$\frac{\beta(d)}{\alpha(d)} < \min_{x \in \mathscr{X}} \min_{c' \neq c(x)} \frac{1}{2}(\delta_{\mathscr{X}}(x,c') - \delta_{\mathscr{X}}(x,c(x))),$$

where $c(x) = \text{argmin}_{c \in \mathscr{C}} \delta_{\mathscr{X}}(x,c)$ denotes the center in $\mathscr{C}$ closest to x. A sufficient condition for the existence of some d satisfying this property is that $\alpha(d)$ is monotone increasing and that $\alpha(d)$ is faster growing than $\beta(d)$. This condition is satisfied for both the random projection and position-ID encoding methods.

Noise Robustness

It is also of interest to consider robustness to noise in the context of encoding Euclidean data. Suppose we have a set of points, $\mathscr{X}$, in $\mathbb{R}^n$, and a distance function of interest $\delta_{\mathscr{X}}(\cdot,\cdot)$ which is preserved *à la* Definition 4. Given an arbitrary point $x \in \mathscr{X}$ we consider a noise model which corrupts $\phi(x)$ to $\phi(x) + \Delta$, where Δ is some unspecified noise process. Along the lines of Section 2.1.2, we say Δ is ρ-bounded if:

$$\max_{x \in \mathscr{X}} |\langle \phi(x), \Delta \rangle| \leq \rho$$

Suppose we wish to ensure the encodings can distinguish between all points at a distance $\leq \varepsilon_1$ from x and all points at a distance $\geq \varepsilon_2$. That is:

$$\|\phi(x)+\Delta-\phi(x')\| < \|\phi(x)+\Delta-\phi(x'')\|$$

for all $x' \in \mathscr{X}$ such that $\delta_{\mathscr{X}}(x,x') \leq \varepsilon_1$ and all $x'' \in \mathscr{X}$ such that $\delta_{\mathscr{X}}(x,x') \geq \varepsilon_2$. We say that such an encoding is $(\varepsilon_1, \varepsilon_2)$-robust.

Theorem 17. *Let $\delta_{\mathscr{X}}$ be a distance function on $\mathscr{X} \subset \mathbb{R}^n$ and suppose ϕ is an embedding preserving $\delta_{\mathscr{X}}$ under the squared Euclidean distance on $\mathscr{H}$ as described in Definition 4. Suppose Δ is ρ-bounded noise. Then ϕ is $(\varepsilon_1, \varepsilon_2)$ robust if:*

$$\rho < \frac{\alpha(d)}{4}(\varepsilon_2-\varepsilon_1) - \frac{\beta(d)}{2}.$$

Proof. Fix a point x whose encoding is corrupted as $\phi(x)+\Delta$. Then for any $x', x'' \in \mathscr{X}$ with $\delta_{\mathscr{X}}(x,x') \leq \varepsilon_1$ and $\delta_{\mathscr{X}}(x,x'') \geq \varepsilon_2$, we have:

$$\begin{aligned}
&\|\phi(x)+\Delta-\phi(x'')\|_2^2 - \|\phi(x)+\Delta-\phi(x')\|_2^2 \\
&= \|\phi(x)-\phi(x'')\|_2^2 - \|\phi(x)-\phi(x')\|_2^2 - 2\langle\phi(x''),\Delta\rangle + 2\langle\phi(x'),\Delta\rangle \\
&\geq \alpha(d)\delta_{\mathscr{X}}(x,x'') - \beta(d) - \alpha(d)\delta_{\mathscr{X}}(x,x') - \beta(d) - 4\rho \\
&\geq \alpha(d)(\varepsilon_2-\varepsilon_1) - 2\beta(d) - 4\rho \ > \ 0,
\end{aligned}$$

as desired. □

As before, we may consider passive and adversarial examples.

Additive White Gaussian Noise. First consider the case that $\mathscr{H} = \mathbb{R}^d$ and $\Delta \sim \mathscr{N}(0, \sigma_\Delta^2 I_d)$; that is, each coordinate of Δ has a Gaussian distribution with mean zero and variance σ_Δ^2. Then, as before, we can note that $\langle\phi(x),\Delta\rangle \sim \mathscr{N}(0, \sigma_\Delta^2\|\phi(x)\|_2^2)$. Then, it is very likely (four standard deviations in the tail of the normal distribution) that $\rho < 4L\sigma_\Delta$, where $L = \max_{x\in\mathscr{X}} \|\phi(x)\|_2$. So then, we have the desired robustness property if:

$$\sigma_\Delta < \frac{\alpha(d)}{16L}(\varepsilon_2 - \varepsilon_1) - \frac{\beta(d)}{8L}$$

Assuming that $\alpha(d)$ is faster growing in d than L and $\beta(d)$, there will exist some encoding dimension for which we can tolerate any given level of noise. In the case of the random projection encoding scheme described above $\alpha(d) = O(d), \beta(d) = O(\sqrt{d})$ and $L = \sqrt{d}$ exactly. And so we can tolerate noise on the order of:

$$\sigma_\Delta \approx \sqrt{d}\,(\varepsilon_2 - \varepsilon_1) - O(1)$$

For the position-ID encoding method, $\alpha(d) = O(d)$, $L = O(\sqrt{nd})$ and $\beta(d) = O(n^2 d\mu)$, and so we can tolerate noise:

$$\sigma_\Delta \approx \sqrt{\frac{d}{n}}((\varepsilon_2 - \varepsilon_1) - O(n^2\mu))$$

Adversarial Noise. We now consider the case that $\mathscr{H} = \{\pm 1\}$, as in the random-projection encoding method, and Δ is noise in which some fraction $\omega \times d$ of coordinates in $\phi(x)$ are maliciously corrupted by an adversary. Since $\|\Delta\|_1 \leq \omega d$, we have, for any $x \in \mathscr{X}$:

$$|\langle \phi(x), \Delta\rangle| \leq \|\phi(x)\|_\infty \|\Delta\|_1 \leq \omega d$$

So then we can tolerate ω on the order of:

$$\omega < \frac{\alpha(d)}{4d}(\varepsilon_2 - \varepsilon_1) - \frac{\beta(d)}{2d}$$

In the case of the random-projection encoding method this boils down to:

$$\omega \approx (\varepsilon_2 - \varepsilon_1) - \frac{1}{\sqrt{d}},$$

meaning the total number of coordinates that can be corrupted is $O(d(\varepsilon_2 - \varepsilon_1))$.

2.4 Learning from HD Data Representations

We now turn to the question of using HD representations in learning algorithms. Our goal is to clarify in what precise sense the HD encoding process can make learning easier. Throughout this discussion, we assume access to a set of n labelled examples $\mathscr{S} = \{(x_i, y_i)\}_{i=1}^{n}$, where x_i lies in $[0,1]^m$ and $y_i \in \mathscr{C}$ is a categorical variable indicating the class label. In general, we are interested in the case that training examples arrive in a streaming, or online, fashion, although our conclusions apply to fixed and finite data as well.

2.4.1 Learning by Bundling

The simplest approach to learning with HD representations is to bundle together the training examples corresponding to each class into a set of exemplars—often referred to as "prototypes"—which are then used for classification [85, 116, 18]. More formally, as described in Section 1.1, we construct the prototype θ_j for the k-th class as:

$$\theta_j = \bigoplus_{i \text{ s.t. } y_i = j} \phi(x_i)$$

and then assign a class label for some "query" point x_q as:

$$\hat{y} = \underset{j \in \mathscr{C}}{\operatorname{argmax}} \frac{\langle \theta_j, \phi(x) \rangle}{||\theta_j||} \tag{2.5}$$

This approach bears a strong resemblance to naive Bayes and Fisher's linear discriminant, which are both classic simple statistical procedures for classification [13]. Like these methods, the bundling approach is appealing due to its simplicity. However, it also shares their weaknesses in that it may fail to separate data that is in fact linearly separable.

2.4.2 Learning Arbitrary Linear Separators

Linear separability is one of the most basic types of structure that can aid learning. The theory of linear models is well developed and several simple, neurally plausible, algorithms for learning linear separators are known, for instance, the Perceptron and Winnow [122, 90]. Thus, if our data is linearly separable in low-dimensional space we would like it to remain so after encoding, so that these methods can be applied. We now show formally that preservation of distance is sufficient, under some conditions, to preserve linear separability.

Theorem 18. *Let $\mathscr{X}$ and $\mathscr{X}'$ be two disjoint, closed, and convex sets of points in $\mathbb{R}^n$. Let $p \in \mathscr{X}$ and $q \in \mathscr{X}'$ be the closest pair of points between the two sets. Suppose ϕ preserves L2 distance on $\mathscr{X}$ under the L2 distance on $\mathscr{H}$ in the sense of Definition 4. Then, the function $f(x) = \langle \phi(x), \phi(p) - \phi(q) \rangle - \frac{1}{2}(||\phi(p)||_2^2 - ||\phi(q)||_2^2)$ is positive for all $x \in \mathscr{X}$ and negative for all $x' \in \mathscr{X}'$ provided:*

$$\frac{\beta(d)}{\alpha(d)} < \frac{1}{2}\|p-q\|_2^2.$$

Proof. We first observe:

$$\langle \phi(x), \phi(p) - \phi(q) \rangle - \frac{1}{2}\left(\|\phi(p)\|_2^2 - \|\phi(q)\|_2^2\right) = \frac{1}{2}\|\phi(x) - \phi(q)\|_2^2 - \frac{1}{2}\|\phi(x) - \phi(p)\|_2^2.$$

We may then use Definition 4 to obtain:

$$\begin{aligned} f(x) &= \frac{1}{2}\|\phi(x) - \phi(q)\|_2^2 - \frac{1}{2}\|\phi(x) - \phi(p)\|_2^2 \\ &\geq \frac{\alpha(d)}{2}\|x-q\|_2^2 - \frac{\alpha(d)}{2}\|x-p\|_2^2 - \beta(d) \\ &= \alpha(d)\left(\langle x, p-q \rangle - \frac{1}{2}\left(\|p\|_2^2 - \|q\|_2^2\right)\right) - \beta(d). \end{aligned}$$

By a standard proof of the hyperplane separation theorem (e.g., Section 2.5.1 of [15]),

$$\langle x, p-q\rangle - \frac{1}{2}(\|p\|_2^2 - \|q\|_2^2) \geq \frac{1}{2}\|p-q\|_2^2$$

for any $x \in \mathscr{X}$, and thus $f(x) > 0$ if

$$\frac{\beta(d)}{\alpha(d)} < \frac{1}{2}\|p-q\|_2^2.$$

the proof for $x \in \mathscr{X}'$ is analogous. □

A natural question is whether a linear separator on the HD representation can capture a *nonlinear* decision boundary on the original data? The connection with kernel methods discussed in Section 2.3.2 presents one avenue for rigorously addressing this question. As noted there, the encoding function can sometimes be interpreted as approximating the feature map of a kernel, which in turn can be used to linearize learning problems in some settings [135].

Learning Sparse Classifiers on Random Projection Encodings

The random projection encoding method can be seen to lead to representations that are *sparse* in the sense that a subset of just $k \ll d$ coordinates suffice for determining the class label. This setting accords naturally with the Winnow algorithm [90] which is known to make on the order of $k \log d$ mistakes when the target function class is a linear function of $k \leq d$ variables. This can offer substantially faster convergence than the Perceptron when the margin is small. Curiously, while the Perceptron algorithm is commonly used in the HD community, we are unaware of any work using Winnow for learning.

Theorem 19. *Let $\mathscr{X}$ and $\mathscr{X}'$ be two sets of points supported on the n-dimensional unit sphere and separated by a unit-norm hyperplane w with margin $\gamma = \min_{x \in \mathscr{X}} |\langle x, w\rangle|$. Let $M \in \mathbb{R}^{d \times n}$ be a matrix whose rows are sampled from the uniform distribution over the n-dimensional unit-sphere. Define the encoding of a point x by $\phi(x) = Mx$. With high probability, $\mathscr{X}$ and $\mathscr{X}'$ are linearly* *separable using just k coordinates in the encoded space, provided:*

$$d = \Omega\left(k \exp\left(\frac{n}{2k\gamma^2}\right)\right).$$

To prove the theorem we first use the following simple Lemma:

Lemma 20. *Suppose there exists a row $M^{(i)}$ of the projection matrix such that $\langle M^{(i)}, w\rangle > 1-\gamma^2/2$. Then $\langle M^{(i)}, x\rangle$ is positive for any $x \in \mathscr{X}$ and negative for any $x \in \mathscr{X}'$.*

Proof. The constraint on the dot product of $M^{(i)}$ and w implies $\|M^{(i)} - w\|^2 = \|M^{(i)}\|^2 + \|w\|^2 - 2\langle M^{(i)}, w\rangle < \gamma^2$. Thus for any $x \in \mathscr{X}$,

$$\langle M^{(i)}, x\rangle = \langle w, x\rangle + \langle M^{(i)} - w, x\rangle \geq \gamma + \langle M^{(i)} - w, x\rangle \geq \gamma - \|M^{(i)} - w\| > 0.$$

A similar argument shows that $\langle M^{(i)}, x\rangle$ is negative on $\mathscr{X}'$. $\square$

Unfortunately, the probability of randomly sampling such a direction is tiny, on the order of γ^n. However, we might instead hope to sample k vectors that are weakly correlated with w and exploit their cumulative effect on x. We say a vector $u \in \mathbb{R}^n$ is ρ-correlated with w if $\langle u, w\rangle \geq \rho$. We are now in a position to prove the theorem.

Proof. For $w \in \mathscr{S}^{n-1}$ and $\rho \in (0,1)$, let $\mathscr{C} = \{u \in S^{n-1} : \langle u, w\rangle \geq \rho\}$ denote the spherical cap of vectors ρ-correlated with w. Suppose we pick vectors $u^{(1)}, \ldots, u^{(k)}$ uniformly at random from $\mathscr{C}$. Then, with probability at least $1/2$:

$$\frac{\langle \sum_j u^{(j)}, w\rangle}{\|\sum_j u^{(j)}\|_2} \geq 1 - \frac{1}{2k\rho^2} \tag{2.6}$$

To see this, note that without loss of generality we may assume $w = e_1$, the first standard basis vector of $\mathbb{R}^n$, and write any $u \in \mathbb{R}^n$ as $u = (u_1, u_R)$: the first coordinate and the remaining $n-1$

coordinates. Now, let $N = \langle \sum_j u^{(j)}, w \rangle = \sum_j u_1^{(j)} \geq k\rho$. Then:

$$\begin{aligned}
\left\| \sum_j u^{(j)} \right\|_2^2 &= \left(\sum_j u_1^{(j)} \right)^2 + \left\| \sum_j u_R^{(j)} \right\|_2^2 \\
&= N^2 + \sum_j \|u_R^{(j)}\|_2^2 + \sum_{i \neq j} \langle u_R^{(i)}, u_R^{(j)} \rangle \\
&\leq N^2 + k + \sum_{i \neq j} \langle u_R^{(i)}, u_R^{(j)} \rangle.
\end{aligned}$$

The last term has a symmetric distribution around zero over random samplings of the $u^{(j)}$. Thus, with probability $\geq 1/2$, it is ≤ 0, whereupon

$$\frac{\langle \sum_j u^{(j)}, w \rangle}{\| \sum_j u^{(j)} \|_2} \geq \frac{N}{\sqrt{N^2 + k}} \geq 1 - \frac{k}{2N^2} \geq 1 - \frac{1}{2k\rho^2}.$$

To ensure the quantity above is at least $1 - \gamma^2/2$, we must have:

$$\rho^2 \geq \frac{1}{k\gamma^2}.$$

It now remains to compute the probability that a vector $M^{(i)}$ sampled uniformly from $\mathscr{S}^{n-1}$ lies in $\mathscr{C}$, or equivalently, that $M_1^{(i)} \geq \rho$. Noting that we may simulate a random direction on $\mathscr{S}^{n-1}$ by sampling $z \sim \mathscr{N}(0, I_n)$ and normalizing, we obtain the reasonable approximation: $M_1^{(i)} \sim \mathscr{N}(0, 1/n)$. Therefore, the probability that $M_1^{(i)} \geq \rho$ is on the order of $e^{-n\rho^2/2}$. So we need:

$$d = \Omega\left(k \exp\left(\frac{n}{2k\gamma^2} \right) \right)$$

□

In summary, the random projection method in tandem with the Winnow algorithm seems to be well suited to the HD setting, where sparsity can be exploited to simplify learning.

2.5 Conclusion

This chapter develops a formal model for studying foundational properties of HD computing. In particular, we are interested in understanding what kinds of structure in the input data are preserved by the encoding operation, and in quantifying the conditions under which the encodings can be decoded to recover the raw input, even in the presence of noise. Our treatment is roughly divided between encoding methods that are suitable for categorical data and those that are suitable for data in a Euclidean space. To study the former, we introduce a model based on representation using incoherent codes, which formalizes the notion "almost-orthogonal" codewords that has long been a central tenant of HDC [75, 48]. In an interesting departure from prior work, our formalism using incoherence does not actually require any randomness in the encoding: in our telling, incoherence is the important property, and randomness is a convenient mechanism for achieving it. Moreover, in contrast to prior work, which has relied on analyses that hold under specific assumptions on the codewords [105], or rely on asymptotic approximations using limit theorems [48, 53], our approach readily yields rigorous bounds in the finite-dimensional setting.

In the second half of the chapter, we develop similar results for representing Euclidean data. In this setting, one typically asks that dot-products or Euclidean distances in HD-space capture some salient notion of distance on the ambient representation of the data. We compute the implied notion of similarity for several popular kinds of encoding encountered in practice and provide distortion bounds that quantify the relationship between the choice of encoding dimension and the fidelity with which this underlying similarity is preserved in HD space. Finally, we conclude the chapter by exploring some of the implications of similarity preservation for different learning applications. We provide sufficient conditions under which cluster structure and linear separability will be preserved in HD space, and again analyze the robustness of these procedures to noise. On a positive note, this chapter establishes rigorous guarantees that the kinds of simple randomized embedding procedures used in HDC are capable of supporting a fairly diverse range of tasks. However, a potential limitation is that naive implementations of these techniques can encounter significant scalability bottlenecks when the dimension of the underlying data is large, for the simple reason that they, in general, require storing at least one HD "codeword" for each dimension that is present in the data. In the following chapter, we explore methods based on hashing for mitigating this issue.

Chapter 2 contains material from "A Theoretical Perspective on Hyperdimensional Computing," by Anthony Thomas, Sanjoy Dasgupta, and Tajana Rosing as it appeared in the Journal of Artificial Intelligence Research (2021). The dissertation author was the primary investigator and author of this paper.

Chapter 3

Hashing and HDC

Many of the encoding methods discussed in previous chapters can be described using the following basic template: one takes a piece of data $x \in \mathbb{R}^m$ and embeds it into $\mathbb{R}^d$ under a random map: $\phi(x) = f(Mx)$, where $f : \mathbb{R} \to \mathbb{R}$ is some optional non-linearity applied element-wise to $z = Mx$. While simple, and remarkably powerful theoretically (per the results of the previous chapter), simple instantiations of this approach encounter serious limitations when the dimension of the input data is large, for the simple reason that one must store and manipulate M. For example, consider the basic encoding method discussed in Chapter 2 for representing sets in which one assigns each symbol a d-dimensional codeword. In this setting, we may take $x \in \{0,1\}^m$ to be the characteristic vector of the set, and $M \in \mathbb{R}^{d \times m}$. Implementing this method directly on an m-dimensional alphabet would require $O(dm)$ storage, which is prohibitive in the setting that m is large relative to the amount of memory available.

In this chapter, we explore a family of techniques from the literature on streaming algorithms [14, 71] for generating encodings "on-the-fly" using hashing. We focus, in particular, on the use of such procedures to construct representations that can be used in learning applications. Our key contributions are as follows:

- We introduce a formal model, based on ideas introduced in Chapter 2, for comparing different encoding architectures for learning applications, and show that hash-based methods enjoy similar theoretical guarantees to standard techniques based on random codeword generation, while being substantially more efficient.

- We analyze the tradeoff between the encoding dimension, the number and type of hash-functions, and intrinsic properties of the data like the size of the input domain, and provide sufficient conditions under which an important family of learning algorithms based on finding linear separators in HD space will succeed.
- We implement hash-encoding techniques on an FPGA and in an "in-memory" architecture, and evaluate them on a large scale classification problem [34], validating our theory and showing that hash-based encodings offer comparable (or superior) levels of accuracy to existing HDC techniques while being more efficient and scalable.

More broadly, our work opens what we believe will be a fruitful line of connection between the hardware and implementation focused literature on HDC, and the theoretical literature on streaming algorithms.

3.1 Background and Related Work

3.1.1 Problem Formulation and Notation

We are here interested in using HD computing to learn classifiers defined over high-dimensional inputs. That is, we are interested in using $\phi(x)$ to fit classifiers, and in the setting that the "dimension" of $x \in \mathscr{X}$ is very large. In particular, we here focus on data that: (1) contains a mix of categorical and numeric features, (2) is gathered continuously and in a streaming fashion, and (3) where the categorical features are drawn from a large alphabet, say on the order of $10^7 - 10^8$, and may not be known in advance.

Let x_c be a vector of s categorical features, and let x_n be a vector of n numeric features. We assume that each coordinate $x_c^{(i)}$ is drawn from some discrete alphabet $\mathscr{A}^{(i)}$, and, without loss of generality, that $\mathscr{A}^{(i)} \cap \mathscr{A}^{(j)} = \varnothing$. Let $\mathscr{A} = \cup_i \mathscr{A}^{(i)}$, and let $|\mathscr{A}| = m$. In other words, we may think of each x_c as a set of s items drawn from an alphabet of m symbols. Our focus is on learning classifiers using HD encodings of such data. Our chief interest is in developing encoding procedures that are efficient in the setting that dimension of the input data is very large. In particular, we seek procedures whose space and time complexity scale only logarithmically in m, and preferably d. For example, the "codebook" based approach described above fails to satisfy this goal since it requires $O(md)$ storage.

3.1.2 Related Work in HDC

Encoding has long been recognized as a costly step in the HD pipeline and there is a substantial amount of work on improving its efficiency (see: [83] for a review). One family of approaches is based on manipulating, in some fashion, a small number of randomly generated seeds, for instance using shifts/permutations [77, 116] or cellular automata [81]. These approaches can be implemented efficiently in the setting that one wishes to repeatedly generate the same sequence of pseudo-random codewords in the same order, for instance, to generate the "ID" vectors in the "position-ID" encoding method described in Chapter 2. However, they are not suitable for our setting in which the symbols to encode arrive in a non-deterministic order. Work in [45] proposes a "random-access" encoder based on permutation, in which one encodes a symbol as a sequence of $b = O(\log m)$ bits, which reduces time complexity to $O(d \log m)$, and storage to $O(d)$, and could, presumably, be used with a sparse seed to further reduce storage requirements. However, this method requires materializing b intermediate d-dimensional representations during encoding, which becomes a bottleneck. We compared this approach to the hashing-based methods studied here, and find that it is $\approx 101\times$ slower when implemented on an FPGA.

A separate family of work is based on the principle of "sparse distributed representation" (SDR) [72, 89, 49] in which one generates codewords containing just k-non-zero values, and can be stored efficiently in offset form using $O(km)$ memory. There are a variety of techniques proposed for different forms of data including structures [110] and Euclidean vectors [112]. While a distinct improvement over dense representations, the linear scaling of memory use in m still renders these techniques problematic in the setting of very large m considered in this work.

In this vein [143, 84] are perhaps most closely related to our present work and also observe that Bloom-filters can be interpreted as an SDR in which the codewords are constructed using hashing and the bundling operator is the logical `or`. However, the focus of this work is on answering membership queries about a single set, which is substantively different from our work, which is focused on learning classifiers from the encoded representations of many different sets. That is to say, they study the problem: given a single set $\mathscr{S} \subset \mathscr{A}$, and a query a, under what conditions are queries $a \in \phi(\mathscr{S})$ answered correctly, while we are interested in learning a linear separator from the encoded representations of many sets $\phi(\mathscr{S}_1), \phi(\mathscr{S}_2), ...$ In particular, their analysis regarding the choice of encoding dimension and number of hash-functions applies to a specific procedure for decoding, which is not germane to our setting, and cannot even be implemented in it.

One of our main contributions is to develop the theoretical foundations of hash-based and sparse encoding methods for use in learning applications. We give sufficient conditions under which hash-based encoding will succeed for a family of practically relevant approaches to learning in HDC, and formally analyze the tradeoff between the encoding dimension, the number of hash-functions, and intrinsic properties of the data like the size of the categorical alphabet, and a notion of separability between classes on the original version of the data.

3.1.3 Hashing

Hashing is a fundamental tool in computer science for constructing space-efficient representations of data. For a good survey of the technical details of hashing, the reader is referred to [147]. Here, we will simply think of a *hash-function* as a map ψ that takes in a member of some set $\mathscr{A} = \{a_1, ..., a_m\}$, and returns a non-negative integer in some pre-defined range $[d] = 1, ..., d$. One typically views ψ being drawn randomly from a family of functions $\Psi = \{\psi : \mathscr{A} \to [d]\}$. For instance, one might take Ψ to be a parametric family, and then instantiate a particular ψ via a random draw of parameters. Moreover, we typically want the output of ψ to appear random in the sense that $\psi(a_1), ..., \psi(a_m)$ simulates i.i.d. draws from the uniform distribution over $[d]$. This is commonly formulated using the notion of a p-independent family, which may be defined as follows [147, Definition 3.31]:

Definition 5. ***p*-Independent Family** *Let $\mathscr{A}$ be an alphabet of m symbols. A family of hash-functions $\Psi = \{\psi : \mathscr{A} \to [d]\}$ is said to be p-independent if, for all sets $\mathscr{S} = \{a_1, ..., a_p\} \subset \mathscr{A}$ of size $p < m$, and any ψ chosen uniformly at random from Ψ, the random variables $\psi(a_1), ..., \psi(a_p)$ are mutually independent and uniformly distributed in $[d]$.*

There is a long history of using hashing for efficient data representation in the broader machine learning literature [14, 17, 32, 26, 137], which have evolved to become a well established part of the empirical toolkit in machine learning [1, 103]. The encoding techniques we pursue here are in this tradition. Indeed, one of our main contributions is to show how the Bloom filter [14], a canonical hash-based method for representing sets, can be used as an efficient encoding strategy with provable guarantees for learning in HDC.

3.2 Learning Model

The choice of encoding function is typically guided by two overriding considerations: efficiency and accuracy, the former being primarily determined by precision, dimension, and sparsity. In the following section, we introduce a model that allows us to compare different architectures along these axes.

To motivate this, we return to the model for classification described in Chapter 1. Let $\mathscr{X}$ be an input space, and let $\mathscr{Y} = \{c_1, ..., c_k\}$ be a discrete label space. We assume one sees a set of samples $\mathscr{D} = \{(x_1, y_1) ..., (x_n, y_n)\}$, which may be presented as a stream (e.g. one-at-a-time), or as a batch. We then construct a representer (sometimes called a "prototype") for each class as a weighted bundle of $\mathscr{D}$:

$$\theta_j = \bigoplus_i \alpha_{ij} \phi(x_i),$$

where α_{ij} is a weight. As noted previously, popular exemplars of this scheme are to simply bundle together the data of a particular class, in which case $\alpha_{ij} = \mathbb{1}(y_i = c_j)$, and to run the Perceptron algorithm, in which case $\oplus$ is the element-wise sum and $\alpha_{ij} \in \{+1, 0, -1\}$ (after the first pass over the data). One then predicts the label of a point according to: $\hat{y}(x) = \text{argmax}_j \langle \phi(x), \theta_j \rangle$ (where the operands may be normalized as necessary), which can be interpreted as learning a linear "score function" in HD space. In even the simplest case that $|\mathscr{Y}| = 2$, for this procedure to succeed (e.g. correctly label a dataset), it is necessary and sufficient that the data be linearly separable in HD space. Thus, we are interested in identifying conditions under which this will be so.

We here simply ask that the input be representable as points in some inner-product space, and that it be separable in that representation. Separability in HD space can then be achieved provided the encoding function preserves dot-products with respect to the ambient representation of the data. We formalize this as follows via a specialization of Definition 4:

Definition 6. $\Delta(d)$***-Dot-Product Preserving Encoding***. *Let $\mathscr{X}, \mathscr{H}$ be inner-product spaces of dimension m and d respectively. Let ϕ be an encoding function from $\mathscr{X} \to \mathscr{H}$. We say ϕ is dot-product preserving if, for all $x, x' \in \mathscr{X}$:*

$$\langle x, x' \rangle - \Delta(d) \leq \langle \phi(x), \phi(x') \rangle \leq \langle x, x' \rangle + \Delta(d),$$

where $\Delta(d)$ is a noise term that depends on d.

Intuitively, randomness in the encoding process adds noise, which generally can be made small by increasing the encoding dimension (d). The precise form of $\Delta(d)$ depends on the encoding method in question. In general, choosing d to be smaller is desirable computationally, but choosing it to be too small can lead to classification errors. One of our main goals is to understand this tradeoff for different kinds of encoding methods.

We formalize this intuition in the following theorem, which extends Theorem 18 from the previous chapter to cover the case that the input sets are not convex to begin with. We first remind the reader that the *convex hull* of a set $\mathscr{X}$, denoted $\texttt{conv}(\mathscr{X})$, is the smallest convex set containing $\mathscr{X}$, or equivalently, the set of all possible convex combinations of points in $\mathscr{X}$.

Theorem 21. *Let $\phi : \mathbb{R}^m \to \mathscr{H} \subset \mathbb{R}^d$ be a $\Delta(d)$-dot-product preserving encoding, and let $\mathscr{X}, \mathscr{X}' \subset \mathbb{R}^m$ be two sets of points satisfying:*

$$\gamma = \|p - q\|_2^2 > 0,$$

where p, q are the closest pair of points in $\texttt{conv}(\mathscr{X})$ and $\texttt{conv}(\mathscr{X}')$ respectively. Then, there exists $\theta \in \mathscr{H}$ and $\nu \in \mathbb{R}$, such that $\langle \theta, \phi(x) \rangle + \nu$ is a valid separator in $\mathscr{H}$, provided $\Delta(d) < \gamma/6$.

Proof. Let $p \in \texttt{conv}(\mathscr{X})$ and $q \in \texttt{conv}(\mathscr{X}')$ be the closest pair of points on the convex hulls of $\mathscr{X}$ and $\mathscr{X}'$. By Caratheodry's theorem, there exists a set of $n \leq m+1$ points $\{x_1, ..., x_n\} \subset \mathscr{X}$, and weights $\{\alpha_1, ..., \alpha_n\} \subset \mathbb{R}^n$, such that:

$$p = \sum_{i=1}^{n} \alpha_i x_i, \text{ and } \sum_{i=1}^{n} \alpha_i = 1.$$

Likewise, there exists a set of $n' \leq m+1$ points, $\{x'_1, ..., x'_{n'}\} \subset \mathscr{X}'$ and $\{\beta_1, ..., \beta_n\} \subset \mathbb{R}^n$, such that:

$$q = \sum_{i=1}^{n'} \beta_i x'_i, \text{ and } \sum_{i=1}^{n'} \beta_i = 1.$$

Now, let us define:

$$\phi(p) = \sum_{i=1}^{n} \alpha_i \phi(x_i), \text{ and } \phi(q) = \sum_{i=1}^{n'} \beta_i \phi(x'_i).$$

Applying definition 6, we first observe that:

$$\begin{aligned}
\|\phi(p)\|_2^2 = \langle \phi(p), \phi(p) \rangle &= \sum_{ij} \alpha_i \alpha_j \langle \phi(x_i), \phi(x'_j) \rangle \\
&\geq \sum_{ij} \alpha_i \alpha_j \left(\langle x_i, x_j \rangle - \Delta(d) \right) \\
&= \|p\|_2^2 - \Delta(d).
\end{aligned}$$

Analogously, $\|\phi(q)\|_2^2 \leq \|q\|_2^2 + \Delta(d)$. Let us define:

$$\theta = \phi(p) - \phi(q), \text{ and } \nu = -\frac{1}{2}\left(\|\phi(p)\|_2^2 - \|\phi(q)\|_2^2\right).$$

Then, fixing some arbitrary $x_o \in \mathscr{X}$, we can see:

$$\begin{aligned} \langle \phi(x_o), \theta \rangle + \nu &= \sum_{i=1}^{n} \alpha_i \langle \phi(x_o), \phi(x_i) \rangle \ldots \\ &- \sum_{i=1}^{n'} \beta_i \langle \phi(x_o), \phi(x_i') \rangle - \frac{1}{2}\left(\|\phi(p)\|_2^2 - \|\phi(q)\|_2^2\right) \\ &\geq \sum_i \alpha_i (\langle x_i, x_o \rangle - \Delta(d)) \ldots \\ &- \sum_i \beta_i \langle x_i', x_o \rangle + \Delta(d)) - \frac{1}{2}\left(\|p\|_2^2 - \|q\|_2^2 + 2\Delta(d)\right) \\ &= \langle x_o, p - q \rangle - \frac{1}{2}(\|p\|_2^2 - \|q\|_2^2) - 3\Delta(d). \end{aligned}$$

By a standard proof of the Hyperplane Separation Theorem [15]:

$$\begin{aligned} &\langle x_o, p - q \rangle - \frac{1}{2}(\|p\|_2^2 - \|q\|_2^2) \\ &\geq \frac{1}{2}\|p - q\|_2^2, \text{ for all } x_o \in \texttt{conv}(\mathscr{X}). \end{aligned}$$

Therefore, we conclude:

$$\langle \phi(x_o), \theta \rangle + \nu > 0, \text{ for all } x_o \in \texttt{conv}(\mathscr{X}),$$

provided:

$$\Delta(d) \leq \frac{\gamma}{6}.$$

The proof for $x_o' \in \mathscr{X}'$ is analogous. □

The parameter γ quantifies how well separated the two sets are to begin with. If γ is large, then we can tolerate more distortion in the encoding, which generally means that d can be smaller. Provided a linear separator exists, there are many suitable algorithms for obtaining one (e.g. Percetrons, SVMs, logistic-regression, etc...). The theorem can be weakened to hold in a pointwise sense, that is for any fixed, but arbitrary, x, in exchange for only requiring $\Delta(d)$-dot product preservation to hold with respect to a fixed set of at most $2m+3 = O(m)$ points.

In the remainder, we will explore specific constructions of ϕ that satisfy Definition 6, for both categorical and numeric data and characterize their relative strengths and weaknesses.

3.3 Encoding High-Dimensional Categorical Data

In the following section, we compare methods for encoding high-dimensional categorical data. We first address the standard method based on generating encodings by random sampling, and then turn to hash-based methods which address some shortcomings of this approach. We show that both methods preserve dot-products with respect to the same underlying embedding, and so both guarantee separability under the same conditions, but that hash-encodings can achieve this guarantee in a far more efficient fashion.

3.3.1 Generating Dense Codes by Sampling

Let $x_c = \{a_1, ..., a_s\}$ be a categorical feature vector with s components, and let $b(x_c) \in \{0,1\}^m$ be the characteristic vector for x_c. That is, the location of the non-zero elements in $b(x_c)$ encode the identity of the symbols in $x_c \subset \mathscr{A}$. The conventional approach to encoding such data in HDC is to simply assign each symbol in the alphabet a encoding generated by random sampling from some distribution over $\mathscr{H}$ [75, 48, 143]. For instance, one might choose: $\phi(a) \sim \text{Unif}(\{+1,-1\}^d)$, for all $a \in \mathscr{A}$. That is to say, each coordinate is sampled independently from the uniform distribution over $\{+1,-1\}$. To generate the encoding for a feature vector, we simply bundle together the encodings for each constituent symbol:

$$\phi(x_c) = \bigoplus_{a \in x_c} \phi(a). \tag{3.1}$$

In the following, let us take $\oplus$ to be the element-wise sum. Then, it can be shown that this scheme satisfies Definition 6 with respect to $\mathscr{X}$ the space of all characteristic vectors encoding sets of size s drawn from $\mathscr{A}$. That is to say, fitting linear separators on $\phi(x_c)$ will succeed if the data is separable with respect to $b(x_c)$. We formalize this in the following Theorem:

Theorem 22. *Let $\phi(a) \sim$ Unif$(\{\pm 1\}^d)$ for all $a \in \mathscr{A}$, and let $\phi(x_c)$ be as defined in Equation 3.1, where the bundling operator is the element-wise sum. Then, with probability at least $1-\delta$:*

$$\left| \frac{1}{d} \langle \phi(x_c), \phi(x_c') \rangle - \langle b(x_c), b(x_c') \rangle \right| \leq O\left(\sqrt{\frac{s^3}{d} \log \frac{m}{\delta}} \right),$$

for all x_c, x_c'.

The proof uses the following Bernstein inequality which may be found in [150, Theorem 2.8.4]:

Theorem 23. ***(Bernstein's Inequality)*** *Let $X_1, ..., X_n$ be a collection of independent mean-zero random variables such that $|X_i| \leq K$ for all i. Then, for every $t > 0$:*

$$\Pr\left(\left| \sum_{i=1}^{n} X_i \right| \geq t \right) \leq 2 \exp\left(-\frac{t^2/2}{\sigma^2 + Kt/3} \right),$$

where $\sigma^2 = \sum_{i=1}^{n} \texttt{var}(X_i)$.

We may now prove the main result:

Proof. of Theorem 22 Fix some pair $x_c, x_c' \subset \mathscr{A}$, let $\phi(x)_i$ denote the i-th coordinate of $\phi(x)$, and let $\mathscr{I} = x_c \cap x_c'$. Then, for any $i \in [d]$:

$$\phi(x_c)_i \phi(x_c')_i = \sum_{a \in \mathscr{I}} \phi(a)_i^2 + 2 \sum_{a,a' \in \binom{\mathscr{I}}{2}} \phi(a)_i \phi(a')_i + \dots$$

$$\dots \sum_{\substack{a \in \mathscr{I} \\ a' \in (x_c \cup x_c') \setminus \mathscr{I}}} \phi(a)_i \phi(a')_i + \sum_{\substack{a \in x_c \setminus \mathscr{I} \\ x_c' \in x_c' \setminus \mathscr{I}}} \phi(a)_i \phi(a')_i$$

$$= |x_c \cap x_c'| + Z_i,$$

where $\binom{\mathscr{I}}{2}$ denotes the set of all distinct pairs of symbols in $\mathscr{I}$. Noting that, for any distinct pair $a \neq a' \in \mathscr{A}$, $\mathbb{E}[\phi(a)_i \phi(a)_i'] = 0$, we conclude:

$$\mathbb{E}[\phi(x_c)_i \phi(x_c')_i] = |x_c \cap x_c'|.$$

Now it remains to show concentration around this value. Let us consider the centered random variable $Z_i = \phi(x_c)_i \phi(x_c')_i - |x_c \cap x_c'|$. Since the terms in Z_i are at least pairwise independent, we may decompose the variance over the sum as:

$$\mathtt{var}(Z_i) = 4 \sum_{a,a' \in \binom{\mathscr{I}}{2}} \mathtt{var}(\phi(a)_i \phi(a')_i) + \sum_{\substack{a \in \mathscr{I} \\ a' \in (x_c \cup x_c') \setminus \mathscr{I}}} \mathtt{var}(\phi(a)_i \phi(a')_i) + \dots$$

$$\dots \sum_{\substack{a \in x_c \setminus \mathscr{I} \\ x_c' \in x_c' \setminus \mathscr{I}}} \mathtt{var}(\phi(a)_i \phi(a')_i) \leq 4s^2,$$

since, for any distinct a, a', $\mathtt{var}(\phi(a)_i \phi(a')_i) = 1$, and there are at most s^2 terms in the sum. Moreover, $|Z_i| \leq 2s^2$, and so, by Bernstein's inequality, for any $t > 0$:

$$\Pr\left(\left| \sum_{i=1}^{d} Z_i \right| \geq dt \right) \leq 2 \exp\left(-\frac{d^2 t^2 / 2}{4ds^2 + 2s^2 dt/3} \right)$$

$$= 2 \exp\left(-\frac{dt^2/2}{2s^2(2 + t/3)} \right).$$

To guarantee this quantity is at most $\varepsilon > 0$, it is sufficient to take:

$$t \geq \max\left\{\sqrt{\frac{16s^2}{d}\log\frac{2}{\varepsilon}}, \frac{8s^2}{3d}\log\frac{2}{\varepsilon}\right\}.$$

The result follows by union-bounding over all $\binom{\binom{m}{s}}{2} < m^{2s}/2$ pairs of sets of size s, whereupon we may set $\varepsilon = 2\delta/m^{2s}$, in which case $\log(2/\varepsilon) \leq 2s\log(m/\delta)$. Since either case in the max above implies we must restrict attention to the regime in which $d = \Omega((s^3 \log m)/d)$, we conclude that, with probability at least $1-\delta$:

$$\frac{1}{d}\langle\phi(x_c), \phi(x_c')\rangle - |x_c \cap x_c'| \leq 4\sqrt{\frac{2s^3}{d}\log\frac{m}{\delta}},$$

as claimed. The lower bound is analogous. $\square$

The bound implies that $d = O((s^3 \log m)/\gamma^2)$ is sufficient to achieve the strong, uniform version of Theorem 21, and $d = O((s^2 \log m)/\gamma^2)$ to achieve the weaker pointwise version. This result strengthens a Theorem of the previous chapter, which yields the same conclusion, but with significantly weaker bounds that imply quartic scaling of d with s.

3.3.2 Hashing Methods for Encoding Categorical Data

The technique described above is inefficient because it requires us to materialize and store a large $d \times m$ dimensional codebook. We now show that hashing based techniques can recover the same formal guarantees, while being substantially more efficient.

Let $\psi_1, ..., \psi_k$ be independent hash-functions from $\mathscr{A} \to [d]$, where $[d]$ denotes the integers $1, 2, ..., d$, and define the encoding $\phi : \mathscr{A} \to \{0,1\}^d$ coordinate-wise by

$$\phi(a)_i = \mathbb{1}(\exists\, j \text{ s.t. } \psi_j(a) = i). \tag{3.2}$$

The embedding of a feature vector is defined, coordinate-wise, as: $\phi(x_c)_i = \max_{a \in x_c} \phi(a)_i$. We note that is it not necessary to actually materialize the $\phi(a)$, we simply state the encoding in this way for conceptual analogy to the previous method. This encoding scheme is, in fact, the same as is in Bloom-filters, a canonical approximate data structure for representing sets [14]. However, the Bloom-filter also relies on a specific decoding scheme, which is neither necessary, nor implementable in our setting. So, while the encodings are constructed in a similar fashion, they are used in a completely different way than is addressed by the standard analyses of Bloom filters.

Critically, for our purposes, the dot-product between the encodings of two sets can be used to estimate the size of their intersection. This is important because it means we do not need to decode the filter to use it for learning. While this general fact is known (see, for instance, [16]), we are unaware of analysis providing error bounds of the form given in Theorem 22. We provide such bounds in the following Theorem:

Theorem 24. *Let x_c, x'_c be sets, each of size s, drawn from an alphabet $\mathscr{A}$ of size m. Let $\phi(x), \phi(x')$ be as defined in Equation 3.2, where we assume the hash-functions $\psi_1, \ldots, \psi_k$ are drawn uniformly at random from a $2s$-independent family. Then, for all x_c, x'_c, with probability at least $1-\delta$:*

$$\left|\frac{1}{k}\langle\phi(x_c), \phi(x'_c)\rangle - \langle b(x_c), b(x'_c)\rangle - \frac{s^2k}{2d}\right| \leq O\left(\max\left\{\sqrt{\frac{s^3}{d}\log\frac{m}{\delta}}, \frac{s}{k}\log\frac{m}{\delta}\right\}\right).$$

To prove the Theorem, we will require the following Lemma:

Lemma 25. *Let $\mathscr{A} = \{a_1, \ldots, a_m\}$ be an alphabet of size m, let $\psi_1, \ldots, \psi_k$ be a set of k hash-functions $\mathscr{A} \to [d]$ drawn uniformly at random from an s-independent family, and let $\mathscr{X} \subset \mathscr{A}$ be any set of $s < m$-symbols drawn from $\mathscr{A}$. Let Z be the number of distinct values in the k hashes of s symbols. Then, with probability at least $1-\delta$:*

$$Z \geq sk - \frac{s^2k^2}{2d} - \max\left\{\sqrt{\frac{2s^3k^2}{d}\log\frac{m}{\delta}}, \frac{4s}{3}\log\frac{m}{\delta}\right\},$$

for all sets $\mathscr{X} \subset \mathscr{A}$ of size s.

The proof of the Lemma makes use of the following martingale form of Bernstein's inequality, which is due to [50, 22]:

Theorem 26. *Let $L_1, L_2, ...$ be a sequence of random variables, $0 \leq L_i \leq 1$. Define the bounded martingale difference sequence $V_i = \mathbb{E}[L_i \,|\, L_1, ..., L_{i-1}] - L_i$, and the associated martingale $S_n = \sum_{i=1}^{n} V_i$, with conditional variance $K_n = \sum_{i=1}^{n} \mathtt{var}(L_i | L_1, ..., L_{i-1})$. Then, for all $t > 0$:*

$$\Pr(S_n \geq t) \leq \exp\left(-\frac{t^2/2}{K_n + 2t/3}\right).$$

We now prove Lemma 25:

Proof of Lemma 25. Fix some $\mathscr{X} \subset \mathscr{A}$ of size s, and let $X_1, ..., X_{sk}$ denote the outcomes of hashing the symbols in $\mathscr{X}$ in some arbitrary order. Now let us define the random variable:

$$Z_i = \begin{cases} 1 & \text{if } X_i \neq X_1, X_i \neq X_2, ..., X_i \neq X_{i-1} \\ 0 & \text{otherwise.} \end{cases}$$

That is, Z_i is $+1$ if X_i is distinct from all previous values, and 0 otherwise. Then, the number of unique values amongst $X_1, ..., X_{sk}$ is given by:

$$Z = \sum_{i=1}^{sk} Z_i.$$

Now let $\mathscr{F}_i = \sigma(Z_1, ..., Z_i)$ be the filtration generated by $Z_1, ..., Z_i$, and note that:

$$\mathbb{E}[Z_i | \mathscr{F}_{i-1}] \geq 1 - \frac{i-1}{d},$$

since probability that X_i is distinct from all previous values is lower-bounded in the case that all

$X_1, .., X_{i-1}$ were distinct. Moreover, since Z_i is Bernoulli:

$$\mathtt{var}(Z_i|\mathscr{F}_{i-1}) \leq \frac{i-1}{d}.$$

Let us define:

$$K = \sum_{i=1}^{sk} \mathtt{var}(Z_i|\mathscr{F}_{i-1}) \leq \frac{sk(sk-1)}{2d} \leq \frac{s^2k^2}{2d}.$$

Now, form the martingale difference sequence $V_i = \mathbb{E}[Z_i|\mathscr{F}_{i-1}] - Z_i$. Then, we may apply the aforementioned martingale Bernstein inequality to conclude that, for any $t > 0$:

$$\begin{aligned}
\Pr\left(\sum_i V_i \geq t\right) &= \Pr\left(Z \leq \sum_i \mathbb{E}[Z_i|\mathscr{F}_{i-1}] - t\right) \\
&\leq \exp\left(-\frac{t^2}{2K + 2t/3}\right) \\
&\leq \exp\left(-\frac{t^2}{\frac{2s^2k^2}{d} + 2t/3}\right).
\end{aligned}$$

To guarantee this quantity is at most ε, it is sufficient to take:

$$t \geq \max\left\{\sqrt{\frac{2s^2k^2}{d}\log(1/\varepsilon)}, \frac{4}{3}\log(1/\varepsilon),\right\}$$

from which the result follows by a union bound over all $\binom{m}{s}$ possible sets of size s, whereupon $\log(1/\varepsilon) \leq s\log(m/\delta)$, and observing that:

$$\sum_i \mathbb{E}[Z_i|\mathscr{F}_{i-1}] \geq sk - \frac{s^2k^2}{2d}.$$

□

The main Theorem follows as a corollary of the previous result:

Proof of Theorem 24. Fix some x_c, x_c', and, as in Lemma 25, let Z denote the number of unique values among the hashes of the symbols in $|x_c \cup x'_c|$. Suppose that $Z = |x_c \cup x'_c|$ – that is, the hashes of all symbols in $x_c \cup x'_c$ are unique. Then, by construction: $\langle \phi(x_c), \phi(x'_c) \rangle = k|x_c \cap x'_c|$. However, in general, if some of the hashes are non-unique then we may either over or under-count the intersection. More formally,:

$$k|x_c \cap x'_c| - \Delta \leq \langle \phi(x_c), \phi(x'_c) \rangle \leq k|x_c \cap x'_c| + \Delta',$$

where $\Delta, \Delta' \leq k|x_c \cup x'_c| - Z$. That is, the error term simply counts the number of non-unique hashes amongst the symbols in $x_c \cup x'_c$.

It now remains to bound this quantity. Noting that $s \leq |x_c \cup x'_c| \leq 2s$, we may apply Lemma 25 to conclude that, with probability at least $1 - \delta$:

$$Z \geq ks - \frac{s^2k^2}{2d} - \max\left\{ \sqrt{\frac{2s^3k^2}{d} \log \frac{m}{\delta}}, \frac{4s}{3} \log \frac{m}{\delta} \right\},$$

and therefore:

$$\Delta, \Delta' \leq \frac{s^2k^2}{2d} + \max\left\{ \sqrt{\frac{2s^3k^2}{d} \log \frac{m}{\delta}}, \frac{4s}{3} \log \frac{m}{\delta} \right\}.$$

Putting everything together, we conclude that, with probability at least $1 - \delta$:

$$\begin{aligned} \frac{1}{k} \langle \phi(x_c), \phi(x'_c) \rangle \leq |x_c \cap x'_c| &+ \frac{s^2k}{2d} \\ &+ \max\left\{ \sqrt{\frac{2s^3}{d} \log \frac{m}{\delta}}, \frac{4s}{3k} \log \frac{m}{\delta} \right\}. \end{aligned}$$

The lower bound is analogous. □

The theorem implies that it is sufficient to take $k = O((s \log m)/\gamma)$ and $d = O((s^3 \log m)/\gamma^2)$ to achieve the uniform version of Theorem 21. To achieve the weaker pointwise result, it is sufficient to take $k = O(\log(m)/\gamma)$, and $d = O((s^2 \log m)/\gamma)$. And so we are able to achieve the same guarantees as in the dense case above, but by only evaluating a small number of hash-functions that depends just logarithmically on the alphabet size. The hash-functions can be described using $O(s \log m)$ memory, and so the memory requirements are $O(s \log m/\gamma)$ which is a factor of s^2 better than storing a d dimension seed codeword as is required in permutation based methods. This can be improved upon further by using pairwise independent hash-functions, which we find sufficient in practice and require just $O(\log m)$ bits to store [147].

3.4 Methods for Encoding Numeric Data

We now turn to methods for encoding numeric data. We here focus on a general family of techniques based on *random projection*. There are other ways to encode numeric vector data (see [143, 83] for detail), but random projections are very general, and have theoretically appealing properties, and so we focus on them here. As in the case of categorical data (Section 3.3), we will consider methods yielding both dense and sparse encodings. Random projection methods are capable of capture very diverse notions of structure in data, and the resulting representations can have powerful properties for learning. For instance, in certain cases, linear separators on random-projection encoded data can capture nonlinear decision boundaries on the original version of the data [119, 113]. We here focus on two instances of this method that are of particular relevance in HDC.

3.4.1 Generating Dense Codes by Sampling

Random projection encoding techniques are generally well known in HDC (see [83] and the references therein). For completeness, we briefly restate one method that can be used to produce dot-product preserving embeddings. Denote by $\mathscr{S}^{n-1}$ the unit-sphere in n-dimensions. Let $x_n \in \mathscr{S}^{n-1}$ be a point to encode, and let $M \in \mathbb{R}^{d \times n}$ be a matrix whose rows are sampled from the uniform distribution over $\mathscr{S}^{n-1}$. Now let $\phi(x_n) = \text{sign}(Mx_n)$, where $\text{sign}(u)$ returns $+1$ if $u \geq 0$, and -1 otherwise. Then, one can show that, to a first order approximation, for any

$x_n, x_n' \in \mathscr{S}^{n-1}$ [143, Corollary 19]:

$$\frac{2}{\pi}\langle x_n, x_n' \rangle - O\left(\frac{1}{\sqrt{d}}\right) \le \frac{1}{d}\langle \phi(x_n), \phi(x_n') \rangle \le \frac{2}{\pi}(\langle x_n, x_n' \rangle) + O\left(\frac{1}{\sqrt{d}}\right), \tag{3.3}$$

We remark that the codes generated using this method can be regarded as a form of "locality-sensitive-hash" in which the collision probability captures a distance function of interest (in this case, the angular distance) on the original data [5]. Note as well that the distortion doesn't depend on the ambient dimension of the data (n), which is appealing when n is very large. One sometimes also quantizes M to be low-precision and/or sparse [108, 109, 107, 65]

3.4.2 Generating Dense Codes by Hashing

As was the case with categorical data, the methods above require one to materialize the embedding matrix $M \in \mathbb{R}^{d \times n}$, which may be infeasible when n (the dimension of the underlying data) is large. To address this issue, we again look to techniques based on hashing. One such approach is the *sparse Johnson-Lindenstrauss transform* (SJLT) [36, 71], which is a procedure for constructing a sparse and low-precision embedding matrix (M) that has an implementation using hashing [71]. The SJLT can be described as a particular construction for a sparse and ternary embedding matrix in which $M_{ij} \in \{-1, 0, +1\}$. Let $k \le d$ be a positive integer, where we assume for simplicity that d is divisible by k. Then, we partition M into d/k blocks, and for each column $j \in 1, ..., n$, randomly set a single row in each block to be either $+1/\sqrt{k}$ or $-1/\sqrt{k}$ with equal probability (which can be done using a hash-function) [30].

Then, it can be shown that the embeddings $\phi(x) = Mx$ preserve pairwise similarities in the sense of Definition 6 (see for instance: [71, 30]). In general, the resulting embeddings are of high-precision (e.g. real numbers). If this is problematic, one can simply compose these encodings with the dense random-projection technique described in Equation 3.3, which is efficient since $d \ll n$ after applying the SJLT.

3.4.3 Generating Sparse Codes

In general, sparse and binary encodings are desirable in that they can be stored more efficiently and can often be used to simplify subsequent computation. A popular approach is to sparsify the output of a random-projection either by thresholding [108, 109, 107, 38], or via a k-winner-take-all operation [72, 8, 37, 38].

The following approach is analyzed in detail in [37, 38]. Let $M \in \mathbb{R}^{d \times n}$, be a matrix whose rows are drawn from $\text{Unif}(\mathscr{S}^{n-1})$, and let $z_i = \langle M^{(i)}, x_n \rangle$, where $M^{(i)}$ is the i-th row of M. Now define:

$$\phi(x_n)_i = \begin{cases} 1 & \text{if } z_i \text{ is in the } k \text{ largest values of } z \\ 0 & \text{otherwise.} \end{cases} \tag{3.4}$$

Intuitively, the $M^{(i)}$ define a set of "receptive-fields" that are activated for inputs x_n that lie within a ball of a certain radius. The precise sense in which this is true is somewhat complex, but intuitively, if $\langle \phi(x_n), \phi(x'_n) \rangle = k_o$, for $k_o \leq k$, then x_n, x'_n must lie in the intersection of the receptive fields of k_o different centers, which constrains their maximum distance. The radii in the receptive fields can be tuned by the choice of d and k. While this technique does not exactly conform to Definition 6, the method is known to have useful properties that make it suitable for learning linear separators [38], and the sparse representations it produces are appealing for practical reasons.

A difficulty with this method is that it requires identifying the k-largest coordinates in $z = Mx_n$, which may be computationally burdensome. However, one can show that a similar locality-preserving property can be satisfied by selecting a threshold t such that, $\Pr(|\langle M^{(i)}, x_n \rangle| \geq t) = k/d$ [38]. We examine this approach empirically in Section 3.7, and show that it offers comparable performance to the dense random projection methods, while enjoying the computational/storage benefits of a sparse binary representation.

3.5 Methods for Combining Numeric and Categorical Encodings

Given embeddings $\phi(x_n), \phi(x_c)$, we need to combine them so as to obtain a final embedding, denoted $\phi(x)$, which will be used to fit the HD model. There are several reasonable ways to do this, we outline a few below, which are compared empirically in Section 3.7, and found to yield roughly equivalent results.

3.5.1 Bundling by Concatenation

The final representation is simply the concatenation of $\phi(x_n)$ and $\phi(x_c)$. This approach may be beneficial because it allows one to easily combine encodings of different precision and sparsity levels, and because the encodings may be of different length. This is appropriate, for instance, when there are many more numeric than categorical features (or vice-versa). However, this comes at the expense of a higher final dimension for the encoding, which means that model will have more parameters to estimate. We note that this approach violates a strict interpretation of the "distributed" paradigm that is typically advocated in HDC. However, it is not clear that this has any meaningful practical implications. For instance, one can still establish noise-robustness guarantees using the results of the previous chapter in this setting.

3.5.2 Bundling by Sum

The final representation is the element-wise sum $\phi(x) = \phi(x_n) + \phi(x_c)$. This approach may be desirable because it (1) does not increase the dimension of the final encoding and (2) captures a notion of similarity between the count and categorical encodings:

$$\langle \phi(x_n, c_c), \phi(x_n', x_c') \rangle = \langle \phi(x_n), \phi(x_n') \rangle + \langle \phi(x_n), \phi(x_c') \rangle + \cdots$$

A disadvantage of this approach is that the encodings need to be of the same length, which means that the numeric encodings may need to be of unnecessarily high-dimension to conform with the categorical encodings, or vice-versa. The final representation may also require higher precision to store than either of the inputs.

3.5.3 Bundling by Thresholded Sum

The procedure above can be modified by thresholding the sum to ensure the result is of low-precision. For instance, one could threshold $\phi(x_n, x_c) = \phi(x_n) + \phi(x_c)$ at 1, which accords naturally with binary and sparse embeddings.

3.6 Implementation in Hardware

From an implementation perspective, the HD representations, and the operators used to manipulate them, are appealing because they are amenable to implementation in highly parallel hardware that allows one to process many coordinates simultaneously. The basic ideas of HDC have existed since at least the late 1980's, however, conventional CPU platforms, which have dominated computing for much of the intervening time, exhibit relatively low-levels of parallelism and are ill-suited to fully exploit the distributed nature of HD representations. The last decade has seen substantial advancements in hardware platforms like graphics processing units (GPUs), field-programmable-gate-arrays (FPGAs), and, even more recently, "in-memory" (PIM) architectures. These media exhibit substantially higher levels of parallelism than commodity central-processing-units (CPUs), and HD implementations in such devices can achieve substantial improvements in energy use and latency compared to CPU based implementations [62, 116, 76].

In the following section, we discuss implementation of the hashing based methods introduced in the previous section on FPGA and PIM. We provide some short descriptions of these platforms and their benefits below. The author claims no credit for these implementations, and gratefully acknowledges the assistance of others (as cited at the end of this chapter) for them. They are included here to illustrate that the hashing techniques discussed above are amenable to the kinds of hardware platforms that have driven much of the recent interest in HDC, and offer practically meaningful improvements over the state-of-the-art.

3.6.1 Implementation in FPGA

For our FPGA implementation, we use Xilinx HLS (High-Level Synthesis) which is a C++ based language suitable for FPGAs and supports a variety of hardware-related optimization directives [33]. Our design is flexible in terms of the encoding dimension (d), number of numeric and categorical features (n and s respectively), precision of the embedding matrix elements (M), and degree of hardware parallelism. Higher levels of parallelism are generally desirable, but require more resources, and so the degree of parallelism that can actually be achieved is dictated by the FPGA size and the complexity of the design as determined by the precision and dimension of the HD representations. Hence, our implementation can be reused for different problems and FPGA sizes by changing the parallelism degree and/or dimension.

Categorical Encoding

To encode a categorical input x_c, with k hash-functions ψ_1 to ψ_k, we pass each symbol $a \in x_c$ through all of the hash-functions and use the hash output to set (write) the resulting coordinates in the encoding to one. Since the output of the hash is not known in advance (unlike, e.g., direct indexes of a loop), the FPGA cannot schedule these writes to be parallel. That is, even if $\phi(x_c)$ is split into several partitions, we can only have inter-partition parallelisms, whereas it is possible all the k hash outputs point to a certain partition, making all intra-partition writes sequential. Thus, a basic implementation of sparse hash encoding takes skt_ψ cycles, where t_ψ is the hash latency. To resolve this issue, we partition the categorical encoding vector $\phi(x_c)$ into p partitions, similar to the partitioning of the rows in M. This is, in fact, a standard alternative way to formulate a Bloom filter [16]. Then, we split the hash-functions among these partitions, $q = {}^{k}/_{p}$ hashes per each. This guarantees each partition does not have more than q write operations, so the hash encoding time reduces to $\frac{skt_\psi}{p}$ cycles. We use the Murmur3 hash-function [6] with a `pipeline` directive that helps the function to achieve a throughput of one hash per cycle.

Numeric Encoding

The numeric encoding is simply a matrix-vector multiplication (e.g. Mx_n) using nested C++ loops over M's rows (outer loop) and columns (inner loop). We parallelize the numeric encoding by entirely unrolling the loop over the M's columns (i.e., the inner loop). That is, all n elements of a row in M are multiplied by the numeric elements in x_n simultaneously and the dot product result is accumulated in a pipelined manner, making an effective vector-vector multiplication throughput of one cycle. We partition M column-wise, so that all elements of a row can be read in the same cycle by storing each element in a different on-chip RAM of the FPGA. We also partially parallelize over the rows of M's (outer loop), to the extent allowed by the FPGA's resources. This row-wise unrolling also requires partitioning the rows of M's into different block RAMs but we found the capability of the FPGA synthesis flow limited in automated partitioning of the large number of M rows (d) to a high degree of parallelism. Thus, we manually partitioned M into p coarse partitions P_1 to P_p, and applied a second round of automated partitioning of R rows over each of these smaller partitions. Thus, effectively, $p \times R$ rows are unrolled.

3.6.2 In-Memory Implementation

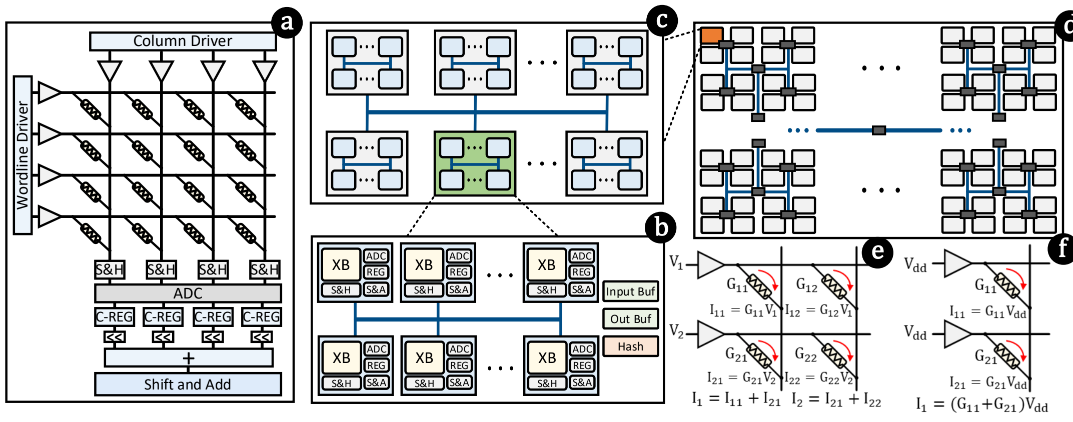


Figure 3.1. Processing in Memory Architecture: (a) ReRAM storage and compute crossbar, (b)–(d) hierarchical tiled organization of the PIM architecture, and (e)–(f) atomic atomic operations.

To make the PIM architecture versatile for other applications, we build upon a general analog PIM design based on ReRAM non-volatile memory crossbar [133]. The crossbar, shown in Fig. 3.1(a), consists of 128 rows and 128 columns (aka bitlines) of cells. The principal functionality of the crossbar is aggregating, sensing and digitizing the current that flows through the bitlines. Essentially, the crossbar can count the number of ones at each bitline. By adding peripheral hardware and decomposing the operations to the bit level, the sensing capability can be used to realize operations such as addition and dot-products between two or more vectors. The columns of a crossbar are further divided into eight vertical lanes, where each lane stores a 16-bit number. This bit-width is typically sufficient in practice for HDC (embeddings and matrix elements) and other learning-oriented use-cases like CNN weights.

A combination of several crossbars forms a cluster, and crossbars in a cluster execute the same instruction (i.e. "single-instruction-multiple-data" (SIMD) processing). A long vector might span across multiple clusters. Intra-cluster data movements, e.g., writing crossbar results to an output buffer, are routed through a shared bus. The crossbars contain input and output registers to temporarily store data, and share a decoding and a hashing unit. Multiple clusters are used to construct a tile, which is shown in Fig. 3.1(c). Tiles are connected through a low-cost network such as H-Tree [51] as global data transfer in PIM applications is expected to be infrequent.

HDC Operations in Memory

A crossbar can sense and store the current passing through the columns using the sample-and-hold (S&H) circuitries. The analog-to-digital converter (ADC) converts the current to digital domain. The ADC operates with higher frequency than the ReRAM operations, hence, a single entity is shared between all columns of a crossbar in a time-multiplexed fashion for a reasonable number of columns. For wider crossbars, higher-frequency ADCs or multiple instances of an ADC can be used.

By applying a voltage of V over an ReRAM cell of conductance G, a current proportional to $V \times G$ passes through the cell. Therefore, a logical 0/1 can be achieved by programming G to 0/1. Summing (bundling) the bits that lie in the same bitline can be done in a single memory

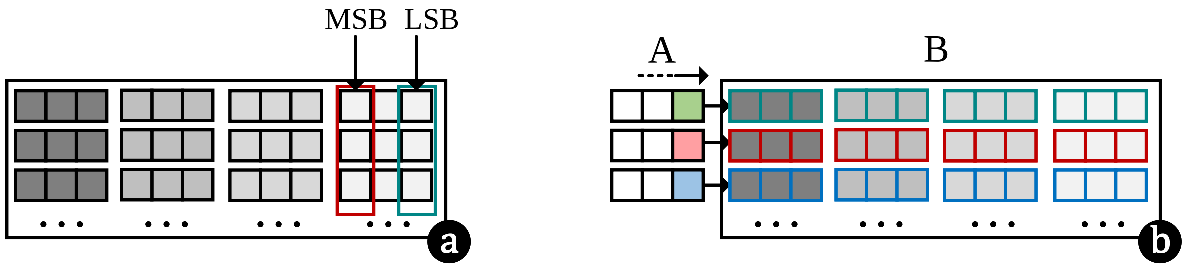


Figure 3.2. HD operations in memory: (a) N-ary addition, and (b) matrix-vector multiplication. Operands may lay in all or a portion of memory columns.

cycle by activating their rows. The result of each bitline is sensed and stored in the column register (C-REG), which holds the data before transferring to other blocks.

Addition of N numbers is performed by placing the numbers in N rows of a lane and performing bit-wise accumulation. Each lane can perform an independent addition as show in Fig. 3.2(a). The result of bit-wise accumulation yields multi-bit partials per bitline, digitized and stored in the column registers of the crossbar. The resulting partial sums are accumulated by passing them through a wired-shift to account for the bit significance of bitlines (i.e., the accumulated result of bit i needs to be shifted left by i indexes), followed by a 16-port tree-adder integrated in each lane. Adding all N numbers of a lane is done in two memory cycles; one to activate the N rows and sample the current, followed by digitizing using the time-multiplexed ADC. Notice that for simple bundling by element-wise sum (as in categorical encoding), each bitline represents a distinct encoding and is independent from the others. Thus, bundling elides further shifting and accumulating the bitline's results.

Dot-products between a pair of vectors $a, b \in \mathbb{R}^n$ is implemented by splitting and applying the elements a in a bit-serial fashion (starting from the least-significant-bit), i.e., $\langle a, b \rangle = \sum_k \sum_i 2^k a_i[k] b_i$, where b is placed vertically on the rows of a lane. As shown in Fig. 3.2(b), if b is a matrix, each of its rows is stored on a different lane and the same a can be applied to multiple lanes of a crossbar simultaneously. Applying the bits of a happens in the form of 0/1 voltage and acts as a `AND` operation with the elements of b. A dot-product between two k-bit vectors or a matrix-vector multiplication takes $k+1$ cycles.

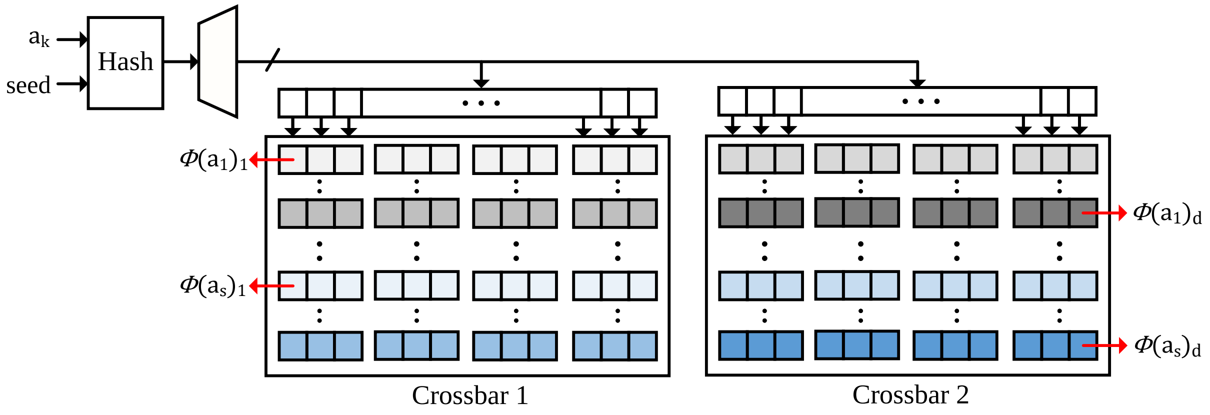


Figure 3.3. Categorical encoding layout in PIM. In this example, two crossbars are allocated to all s required vectors. Each vector is placed on the allocated crossbars in a row-major fashion. The same index of different vectors lay in the same bitline.

In-Memory Sparse Categorical Data Encoding

Fig. 3.3 shows the procedure for sparse hash-encoding of categorical data. First, the number of crossbars to store s categorical binary vectors are determined and allocated. Here each *bit* of the crossbar stores one bit of $\phi(a_i)$. A vector $\phi(a_i)$ has a length of d and spans over all the allocated crossbars in a row-major order, and might be placed in multiple rows of the designated crossbars. Similar to the FPGA implementation, to facilitate writing sparse bits to the vectors, we logically partition each vector and enforce the criterion that only one bit per partition can be one, the index of which is determined by the hash-function. The decoder in the Fig. 3.3 converts the hash-function output to a one-hot signal to write the proper value into the driving register. We process the allocated crossbars row by row, starting from the first row of $\phi(a_1)$. The hash-function $\psi(a_1)$ determines the single bit index of the row (within all crossbars) that needs to be set to 1. Thus, processing each physical memory row within all crossbars takes one memory cycle. By default, we pack all vectors into the minimum number of crossbars. Thus, all the rows of the designated crossbars are filled, and generating the sparse vector takes ~ 128 cycles (one cycle per allocated row). Bundling can then be achieved by activating the requisite rows. We cannot activate all rows at once since a crossbar stores multiple segments of the same vector. In expectation, there are ${}^{128}/_{s}$ different segments of all vectors in a crossbar which need separate bundling cycles. Depending on the input parameters, we might allocate more than minimal crossbars (e.g., to balance the performance of numeric and categorical encoding) which reduces the number of utilized rows of crossbars, and therefore, the encoding time.

In-Memory Numeric Data Encoding

Numeric encoding involves matrix-vector multiplication, which is inherently supported by PIM as explained above. To realize the numeric encoding Mx_n, we first determine the number of crossbars required to fit M. The matrix M then vertically spans the lanes of the allocated crossbars in a row-major order. Since a row of M may not use up all the rows of a crossbar, we can place multiple rows of M within a lane of the crossbar (e.g., M_1 and, say, M_9 are placed in the same lane). These rows, however, need separate iterations to be multiplied with the feature vector to avoid the unwanted aggregation of their current in the column registers. Finally, the outputs of the lanes are transferred to the output register via the shared bus.

3.7 Empirical Evaluation

We evaluate our approach on three classification problems, summarized in Table 3.1. The "language" and "news" tasks are relatively small scale text classification problems which are well-known benchmarks in the HDC literature [76, 96, 117]. In the "language" task, the goal is to identify the language in which a sentence of text is written. In the "news" task, the goal is to identify the topic (e.g. "finance," "technology," etc...) of news articles. For the language identification task, we use the pre-processed data provided by [117], and for the news task, we use the pre-processed data provided by [121]. The conventional approach in the HDC literature is to partition the text into words consisting of n characters (called "n-grams"), which are represented by binding together encodings of each character. This amounts to constructing a representer for each distinct n-gram. A document is then represented as a bundle of encodings of its n-grams. Thus, the implicit number of symbols to represent in these datasets (e.g. what we call the "alphabet size") is the number of n-grams in the text.

In this particular case, the representations of the *n*-grams can be built incrementally, storing encodings only for the 27 primitive characters, and so the storage requirements of even the standard implementation are modest (although we note hashing still achieves about a 210$\times$ reduction in memory use in this case). Our approach simply hashes each *n*-gram directly, but the underlying modeling assumptions are the same (e.g. the unique "tokens" to represent are *n*-grams). These problems are relatively small scale examples, and we include them primarily to verify that our methods are competitive with established techniques on existing benchmarks.

To showcase the benefits of hash-based encoding for truly large-scale problems where such compositionality cannot be exploited, we apply these methods to the "Criteo" click-through-rate prediction dataset [34]. We selected this dataset because it is, to the best of our knowledge, by far the largest publicly available classification dataset, and is generally considered challenging even for sophisticated deep learning based approaches, for which hashing based dimensionality reduction techniques also appear to be popular [41]. Indeed, hashing based methods are well known in the broader literature on this problem [25], although naturally none of this work has explored connections with HDC. We emphasize that it is not our goal to achieve state-of-the-art accuracy on this task; we merely wish to understand how hash-encoding compares to conventional techniques in the HDC literature.

The data contains information about web advertisements displayed to customers along with a binary label indicating whether or not the advertisement was clicked. The data contains 13 numeric features, and 26 categorical features defined over a categorical alphabet of 3.4×10^7 symbols. The goal is to predict whether or not a customer clicked on an ad. The data comes in two sizes: a "full" dataset (about 1 TB on disk - "Criteo (B)" in Table 3.1) covering a month of served ads, and a carefully selected (e.g. non-random) subsample created for a Kaggle challenge (about 10 GB on disk - "Criteo (A)") that covers one week of ads. We primarily focus on the smaller 7-day dataset to keep runtimes tractable when comparing a large number of design choices. We emphasize that the scalability of the hashing based approaches we advocate depends only on the number of features, and the size of the categorical alphabet. Holding these constant,

Table 3.1. Datasets

	Samples	Classes	Alphabet Size (m)
Languages	$242,027$	22	$149,765$
News	$7,674$	8	$19,263$
Criteo (A)	4.6×10^7	2	3.4×10^7
Criteo (B)	4.3×10^9	2	1.9×10^8

Table 3.2. Results on text classification problems. Baseline results are from [76] (Table 2.c). The baseline uses a dense encoding method with $d = 10,000$. Hashing methods $k = 4$ and $d = 10,000$.

	Dense Baseline	**Hashing**
Languages	97.0%	98.9%
News	93.6%	94.1%

the total number of observations/size of the data is irrelevant from the perspective of computation. Thus, from the perspective of understanding scalability, there is little difference between the two datasets.

3.7.1 Results on Text Classification Tasks

For the text classification tasks, we represent each class as superposition of the corresponding training data, that is we set $\theta_c = \sum_{i=1}^{n} \mathbb{1}(y_i = c)\phi(x_i)$, which we then round to binary precision by thresholding at the median of each prototype. Results are reported in Table 3.2. Baseline results are obtained from [76] (Figure 2.c), who use a dense encoding procedure with $d = 10,000$. We tried a number of hash-functions ranging from $k = 1$ to $k = 64$, and found no significant difference. We report numbers for $k = 4$, simply because this was the value that actually maximized accuracy.

3.7.2 Results on Criteo

In the previous section, the weights used to combine together the points belonging to a particular class are constrained to be the same for all points in that class. While simple, this constraint can be overly restrictive for more complex problems, and so we here look to techniques which allow us to learn an adaptive weight α_i for each point: $\theta_c = \sum_{i=1}^{n} \alpha_i \phi(x_i)$. A common approach in the literature on this problem is to model the log-odds of a click as a linear function of the input (e.g. a logistic regression) [25]. We thus also adopt this approach, and fit parameters using mini-batch stochastic-gradient descent (SGD) on the negative log-likelihood of the data. That is, we estimate $\log \frac{p}{1-p} = \theta \cdot \phi(x_n, x_c)$, where $\phi(x_n, x_c) \in \mathscr{H} \subset \mathbb{R}^d$ is the HD encoding, p is the estimated probability that the ad was clicked, and $\theta \in \mathbb{R}^d$ is the vector of parameters to be estimated. Given a new sample (x_n, x_c, y), the update rule in this case becomes $\theta \leftarrow \theta - \alpha\phi(x_n, x_c)$, where $\alpha = \eta(\sigma(\phi(x_n, x_c) \cdot \theta) - y)$, where σ is the logistic-sigmoid [57], and η is the step-size. Thus, as in the case of the standard HDC "retraining" approach based on the perceptron algorithm [62], we model θ as a linear combination of the encodings of training data. Moreover, this approach can also easily be implemented in the online setting in which data is streamed continuously. However, we find that logistic regression delivers substantially improved results compared to the Perceptron (about 5% greater accuracy). The hashing based methods were implemented in CPU using a custom C-Python extension built by modifying the Murmur hash library provided by [132].

Following prior work, we use the first $6/7$ of the data (roughly corresponding to 6 days) for training, and partition the remaining $1/7$ evenly between testing and validation [41]. The validation set is used to tune model parameters like number of hash-functions and encoding dimension and to determine when to stop training the model. Models are validated every $300,000$ records, and we stop training if the loss fails to decrease after 3 consecutive rounds of validation. Again, following standard practice, we assess model performance using the "area under the receiver-operating characteristic curve" (AUC), which better reflects model performance on imbalanced datasets than raw accuracy [41].

Comparing Hash-Based and Random Encoding Methods

Figure 3.4 compares the scalability of encoding methods for categorical data. We here measure the time to encode a batch of $100,000$ observations as the encoding dimension is varied.

Solid lines indicate the sparse encodings generated using the Bloom filter based method described in Section 3.3, and dashed lines indicate encodings generated using random-sampling. We do not include the dense hash methods in this plot because they are dramatically slower and would obfuscate the plot. Our random-encoding technique lazily populates a codebook as new symbols are encountered in the data. Thus, the size of the codebook (and the amount of memory required) will increase as more data is processed. To ensure a fair comparison with our `C` implementation for hash-based encoding, we also implement random-encoding as a custom Python extension written in `C`, avoiding the high overhead of encoding in native Python.

As can be seen in Figure 3.4, random encoding generation rapidly encounters scalability bottlenecks as the volume of data processed increases. This is because the categorical alphabet size scales roughly linearly with the number of observations processed, which necessitates storing an ever larger codebook. At a certain point, the codebook size exceeds available RAM, and the program crashes. This problem can be mitigated by using smaller encodings (potentially at the expense of accuracy), or by using caching schemes which retain only the most frequently accessed encodings in memory. However, such approaches do not resolve the fundamental problem, and come with attendant costs in terms of accuracy, latency, and implementation complexity. The naive hash-based method improves on the situation in some ways, since it does not require actually storing the encodings, but it rapidly becomes bottlenecked by computation when d is large. For instance, with $d = 500$, encoding a single batch of data takes about 36 seconds on a standard CPU machine.

By contrast, in the sparse-hashing based approach, the number of hash-functions remains fixed regardless of the volume of data processed—and the encoding dimension—and so the hash-based encoding methods exhibit constant, high-performance. There is a small overhead from using a larger encoding dimension due to memory allocation, but this is modest. This plot underscores our fundamental observation in this work: encoding techniques that require materializing a codebook simply do not scale to large alphabet sizes. By contrast, hash-based methods offer constant performance independent of the volume of data processed and easily scale to very large data sets.

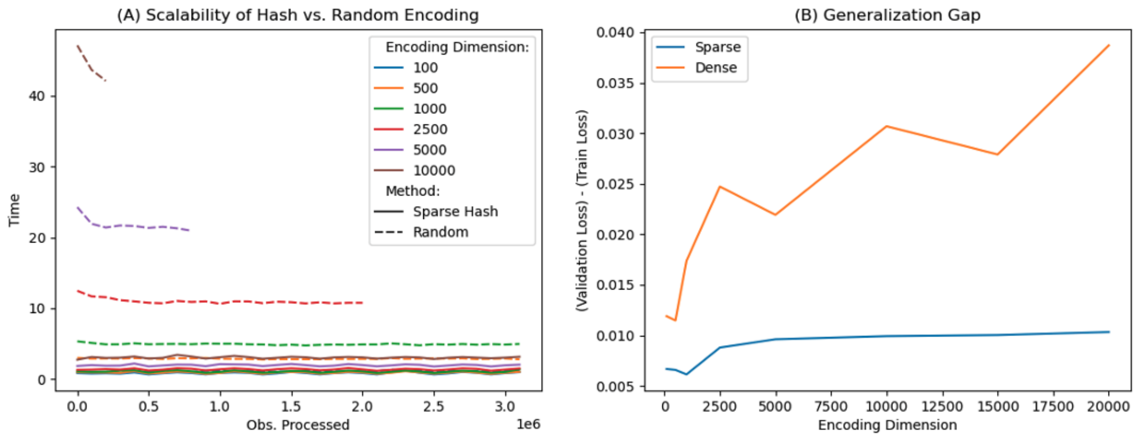


Figure 3.4. Panel (A) plots the time to encoding batches of 100,000 observations using naive random encoding generation, and the sparse hashing-based method of Section 3.3. Panel (B) compares the gap between validation and training loss, for the dense vs. sparse hash based encoding at different values of d_{cat}

Evaluating Hash-Encoding Parameters

We here evaluate the effect of the encoding dimension d_{cat} for categorical data, and the number of hash-functions/sparsity k on model performance. Results are presented in Figure 3.5. For both panels, the numeric encoding method is dense random-projection with fixed $d = 10,000$. The bundling method is concatenation, meaning the final model contains $10,000 + d_{\text{cat}}$ parameters. We partition the test and validation sets into chunks, each consisting of 100,000 samples, and report distributions of model performance, as measured by AUC, as box-plots. The shaded box indicates the 1-st through 3-rd quartile, and the solid line indicates the median. The whisker length is $1.5\times$ the inter-quartile range. The number in the box-plot is the median.

Figure 3.5 (A) compares model AUC as the number of hash-functions is varied, with a constant $d_{\text{cat}} = 10,000$. We find that $k = 4$ delivers the best median test error, but that the difference in performance between $k = 1$ and $k = 100$ is not significant. This is consistent with theoretical results that show error as an increasing function of k for a fixed d. This result is appealing from a practical perspective because it means that one can obtain good performance using a handful of hash-functions. Evaluating the hash-functions is the most expensive part of the Bloom filter based encoding method and so reducing k would be expected to lead to better performance.

To provide context on these results, [41] presents the most recent systematic comparison, to our knowledge, of different deep learning architectures used on this problem and requires $36-540$ million parameters to achieve AUCs of $0.8-0.81$. By contrast, the models in Figure 3.5 contain $\sim 20,000$ (trainable) parameters. We again emphasize that we do not seek to compete with state-of-the-art results on this task. We merely include these comparisons to emphasize that our method falls within the ballpark of results in the literature dedicated specifically to this problem.

Figure 3.5 (B) presents an analogous plot that fixes $k = 4$, and varies the encoding dimension. We here also compare the Bloom filter based encoding method, with the baseline of dense hashing as described in Section 3.3. Again, consistent with theoretical analysis, performance is strictly increasing for both methods as the dimension is increased. Increasing the categorical encoding dimension results in a consistent increase in AUC up to $d_{\text{cat}} \approx 10,000$ at which point the model saturates and increases in AUC become insignificant. We emphasize that the sparse hash-based encoding methods described here are also advantageous because the number of memory accesses needed for an inference computation depends only on k, the number of hash-functions, rather than d the encoding dimension. Accordingly, the increase in accuracy from $d_{\text{cat}} = 500$ to $d_{\text{cat}} = 20,000$ is cheap since the number of hash-functions is held constant.

Interestingly, we find that the sparse hash-encoding method offers markedly better performance for large d_{cat} than the dense baseline. This is welcome news from a practical perspective since sparse encodings are considerably more efficient computationally, but somewhat surprising given that both methods were shown to preserve dot-products in the sense of Theorem 21. We attribute this to a greater propensity for over-fitting for models trained on the dense embeddings, than on the sparse. Figure 3.4 (B) plots the gap between train and validation loss, averaged over the last 10 rounds of validation, for both the sparse and dense encoding methods. The dense encoding method over-fits with increasing severity as d_{count} is increased. On the other hand, over-fitting increases very gradually in the sparse representations. This is because only a tiny fraction of the models parameters ($\approx ks/d$) are updated by any given training example–similar in spirit to dropout regularization in deep neural networks. Over-fitting on dense representations could presumably be addressed by L1/L2 regularization, as is in the LASSO model. However, this introduces another hyper-parameter that must be tuned, which is computationally burdensome. The sparse encoding strategy, by contrast, seems to suffer from only very modest over-fitting without the need for any explicit regularization, and are more computationally efficient.

Comparing Methods for Encoding Numeric Data

Figure 3.6 (A) compares methods for encoding the numeric data. We compare the dense and sparse random-projection based encoding methods described in Equations 3.3 and 3.4 respectively, along with the SJLT described in Section 3.4. To simplify implementation of the SJLT, we relax the construction of [71] and simply instantiate M by drawing each coordinate uniformly at random from the distribution:

$$M_{ij} = \begin{cases} +1 & \text{w.p. } p/2 \\ 0 & \text{w.p. } 1-p \\ -1 & \text{w.p. } p/2. \end{cases} \tag{3.5}$$

We compare the performance with different choices of p (i.e. number of non-zero components). The categorical encoding method is the sparse "Bloom filter" based method, with $d = 10,000$, and $k = 4$. The bundling method for the numeric and categorical encodings is concatenation. The box-and-whisker plots are as described for Figure 3.5.

We compare these approaches against two baselines. The first is to simply omit the numeric data all together, and fit the classifier only on the categorical encodings. The purpose of this baseline is to verify that the count encodings are indeed useful to the classifier. The second is to encode the numeric data using a simple multilayer-perceptron (MLP) style neural network. The MLP contains 4 hidden layers with $512 \times 256 \times 64 \times 16$ hidden units in each layer for a total of $155,984$ parameters. The MLP is trained along with the logistic-regression classifier using SGD.

We find that the MLP and SJLT, with a sparsity parameter of $p = 0.4$ deliver best results, each achieving a median test AUC of 0.77 and outperforming the random-projection based methods that rely on dense projection matrices. The SJLT offers two significant advantages: first, it is instantiated randomly at the start of training, and remains fixed from then on. Second, the coordinates in the embedding matrix–M–are sparse ($\sim 60\%$ zeros), and low-precision. By contrast, the MLP weights are dense and high-precision, and must be trained using back-propagation. A potential disadvantage of the SJLT is that the encodings (e.g. $\phi(x)$) are dense. We find that the sparse random projection method loses just $0.007 - 0.005$ AUC relative to the SJLT to achieve sparsity levels of 1% and 10% respectively in the encodings, but at the expense of needing to store a dense and high-precision embedding matrix. It would be of interest to study ways to combine these methods. That is, to have both sparse embedding matrices and encodings.

3.7.3 Comparing Methods for Bundling Encodings

Figure 3.6 (B) compares the different methods for combining (or "bundling") the encodings of numeric and categorical data described in Section 3.5. The categorical encoding method is the Bloom filter with $d = 10,000$ and $k = 4$, and the numeric encoding method is sparse random projection with $d = 10,000$ and $k = 100$. All three methods yield nearly equivalent performance in terms of AUC. However, bundling by the logical "or" is advantageous from a computational perspective since (1) it does not increase the dimension of the bundled representation, and (2) the final embedding is fully binary.

Table 3.3. Frequency, number of cycles (of each step), and throughput (millions of inputs per second) of the FPGA implementation ($d = 10{,}000$).

	Frequency	$\phi(x_c)$	$\phi(x_n)$	$\langle\theta,\phi(x)\rangle$	$\big(\sigma(\langle\theta,\phi(x)\rangle - y_i\big)\phi(x)$	Throughput (M/sec)
`OR`	130 MHz	31	48	35	34	1.51
`SUM`	122 MHz	57	48	40	34	1.08
`Concat`	150 MHz	31	80	67	66	0.94
`No-Count`	150 MHz	49	–	20	18	2.69

3.7.4 Hardware Evaluation

FPGA Evaluation

We implemented the design using Xilinx Vitis HLS 2021.2 [33] on an Alveo U280 Data Center Accelerator Card. The FPGA is installed in a machine running Ubuntu 18.04 with Intel Gen-11 Core i7-11700K @4.8 GHz and 80 GB of physical memory. We implemented all the three combining techniques, namely thresholded-sum (`OR`), sum (`SUM`), and concatenation (`Concat`), which achieved operating frequencies of 122–150 MHz. We also implemented the `No-Count` encoding that omits the numeric data and works at 150 MHz. We used five manual coarse partitions for the projection matrix rows and vectors (i.e., $p = 5$ as described in Section 3.6.1), followed by a per-partition parallelism degree of $R = 64$ in the `OR` and `SUM` combining methods, which makes an effective parallelism of 320. It means that we can multiply 320 rows of the matrix M with the numeric features per cycle. The total dimensionality of the `Concat` combining is larger ($20K$), so we could set $R = 32$ to avoid routing congestion. The `No-Count` achieved a higher parallelism of $R = 128$ as it uses considerably less resources.

Performance: Table 3.3 reports the cycle count for each of the modules. Since the encoding and update modules work in a dataflow fashion, the total latency is the maximum of encoding (sum of categorical and numeric) and update latency. The entire design is balanced, so the encoding and update modules require a similar amount of time to complete. Combining encodings using the `SUM` approach takes more cycles than the `OR` as the latter only sets a certain subset of coordinates to 1, while in `SUM` encoding, the embeddings are of higher precision. Thus, an extra read per embedding is needed and the next hashes need to wait for the current result since multiple hash outputs might point to the same index. The numeric encoding column $\phi(x_n)$ includes the latency of writing to the dataflow FIFO, as well (the $\phi(x_c)$ column in case of `No-Count`). In the `Concat` combining technique, both parts of the combined vector work in parallel, but we achieved lower ($R = 32$) parallelism due to high resource utilization, so the latency of its stages is higher. On the other hand, `No-Count` is the fastest encoding due to using larger parallelism, but incurs a cost in accuracy as described above. The last column of Table 3.3 report the throughput of FPGA implementation in terms of million inputs per second (both encoding and learning by calculating gradient). The FPGA implementation can process between 939K and 2694K inputs per second.

Resource and Power Consumption: Fig. 3.8 shows the FPGA resource utilization for different combining techniques. `OR` and `SUM` use a similar amount of resources except `SUM` uses slightly more DSPs due to the higher precision of categorical embeddings. `Concat` uses fewer DSPs due to lower parallelism ($R = 32$), meaning that it performs half of the vector-vector multiplication (M and x_n) of the other two. However, LUT and FF utilization of `Concat` is similar to `OR` and `SUM`. Despite using half as much parallelism, the total length of the `Concat` vectors is twice ($d = 20{,}000$), so overall this method uses a similar amount of resources. Finally, `No-Count` does not involve numeric encoding, so it the least amount of DSPs.

The curve in Fig. 3.8 (right y-axis) shows the power consumption of the FPGA, measured using Alveo's real-time power monitoring. The FPGA consumes an idle power of $\sim$24 W which contributes to the major component of the power drain. The resource utilization and operating frequency of different combining approaches is similar, so the total FPGA power hovers around 30 W (minimum 26 W for `No-Count` and maximum 30 W for `OR`).

Encoding and Update (Learning): We compare the FPGA implementation and CPU for end-to-end learning, i.e., encoding followed by an update of model parameters. We have not yet evaluated the learning step in PIM, and leave it to future work to accelerate learning using logistic regression in PIM. We note however that more conventional HD based learning algorithms using bundling can be effectively implemented in PIM [76], but, these approaches do not provide sufficient statistical power in this context. Fig. 3.9(a) compares the end-to-end performance of FPGA and CPU implementations for various combining techniques. The FPGA results are the same reported in Table 3.3. Our FPGA implementation outperforms the CPU by 115× to 163× depending on the method used to bundle the categorical and numeric encodings. Fig. 3.9(b) considers the power consumption and reports the throughput/Watt. The different bundling methods]consume a similar amount of power in each platform. CPU power use hovers around 88 W, while FPGA power ranges from 26 W in `No-Count` to 30 W in the `OR` technique. Accordingly, the FPGA implementation achieves 349× to 508× better throughput/Watt than CPU. With a throughput of 1.51 M/second (for `OR`), each epoch on the large Criteo (B) dataset takes $4.3 \times 10^9/1.51\text{M} = 0.79$ hour on FPGA, costing only $12 \times 0.79 \times 30/1000 \approx 0.30$ cents with a power of 30 W and average electricity price of 12 cents per kWh [3]. In comparison, the CPU costs 1.40 USD.

Table 3.4. PIM performance details. The allocated crossbars are per each input; multiple inputs are being processed in parallel.

	Allocated Crossbars		Utilization Rate		Encoding Cycles		Throughput
	Numeric	Categorical	Numeric	Categorical	Numeric	Categorical	(M/sec)
`OR/SUM`	144	40	91%	41%	81	80	21.97
`No-Count`	–	20	–	81%	–	132	103.41

Comparison to Bit-Serial Re-Materialization

A plausible alternative to the hashing methods discussed above is that of [45], which represents a symbol as a sequence of bits, and only requires storing a single "seed" vector $\mathbf{s} \in \mathscr{H}$, which is repeatedly permuted to encode the bit-sequence representing a symbol $a \in \mathscr{A}$. For instance, for a symbol a represented as a sequence of bits $a = b_1 b_2 b_3$, the encoding is generated via $\phi(a) = \pi_{b_3}(\pi_{b_2}(\pi_{b_1}(\mathbf{s})))$, where π_0, π_1 are hard-wired permutations. As the shuffle operators are hardwired, almost no computation logic is required, which means the bit-serial encoder is always able to achieve high bit-level parallelism. The encoding requires $O\big(d/K \times n\big)$ cycles where d is the HD dimension, K is the bit-level parallelism and n is the number of bits of the input data.

As is noted in [45], this method could also be used with a sparse seed to further reduce memory requirements. When only $\frac{1}{s} \times 100\%$ dimensions are nonzero in the seed, one possible optimization for area utilization is to replace the hardwired shuffle with an index lookup table and reduce the number of process units from K to K/s while still maintaining the same throughput. The index lookup table contains K elements and stores the mapping relationship of the shuffle. The active processed element is changed from K single bits to $K/s\ log(K)$-bits index.
Using $K = 1024, d = 10,000$, we obtain that encoding a single 32-bit categorical feature (as in the Criteo data) requires 382 cycles for the dense implementation and 392 for the sparse alternative (with $s = 128$, the most sparse configuration in [45]), bringing the cycle counts for all 26 16-bit input features to $9,932$ (dense) or $10,192$ (sparse), causing the categorical feature encoding to throttle the throughput of the pipeline. While the sparse configuration requires more cycles, it is able to save 9% of resources compared to dense. Because the shuffle operator is simpler than the hashing operator, the pipeline can achieve 300MHz compared to 150MHz in the hash-based implementation. All told, we find that the overall speedup, for encoding, of the hash-based approach compared to the bit-serial approach of [45] for encoding is $101\times$.

Comparison to Shift-Based Re-Materialization

Another approach to generate random categorical vectors could be by permuting a set of seed vectors, where a seed vector of length d can create d different vectors [77]. Thus, we can associate each categorical feature vector with a certain permutation of a seed vector, so that the total number of required vectors reduces by a factor of d (e.g., 22.6 MB for the Criteo dataset). Selecting the seed vector from the pool of the seeds and then the number of permutations on the selected vector depends on the feature value. A simple approach to realize it to use two hash-functions over the value of the categorical feature, $\psi_1(a)$ and $\psi_2(a)$ to determine the seed and the number of permutations. The permutation is variable within $[0, d)$, so it requires $O(d)$ cycles (note that we cannot have variable permutation in $O(1)$). To improve that, we set the permutation granularity to 16 by choosing permutation steps as $\left(\psi_2(a)\%\frac{d}{16}\right)\times 16$, meaning that a seed vector can be permuted only in multiples of 16. It eases the vector materialization by using $O(\frac{d}{16})$ cycles to permute, which we further improve by using data movement instead of permutation. Once the proper seed vector is read from the FPGA DRAM, we split it into segments of 16 successive bits. We initialize a categorical (level) vector, and the hash value $m = \psi_2(a)\%\frac{d}{16}$ determines the index from which the *bricks* should be written to the level vector. Accordingly, the i^{th} seed brick goes to the brick number of $(m+i)\%\frac{d}{16}$ of the materialized level vector.

With all the explained optimizations, we observed that materialization of each level vector (per each categorical feature), including reading the seed vector from DRAM, takes ∼500 cycles. This overshadows the compute latency, so the performance of previous works (e.g. [127]) will be limited by the vector materialization as they need to store the entire codebook. The throughput of this approach is limited to ∼11,200 inputs/second with the categorical encoding being the bottleneck stage in all combining approaches. Thus, encoding by materialization is 84× slower than our slowest hash-based approach (`Concat`), and 135× slower than our hash-based encoding using `OR` combining.

PIM Evaluation

Table 3.5. PIM components specifications.

Component	Area (μm^2)	Power (μW)	Component	Area (μm^2)	Power (μW)
128×128 array	25	300	Hash	839	8.8
ADC	570	1451	Decoder	26	0.02
DAC (×256)	136	5.4	Router	2209	459
S&H (×128)	5.0	1.0			
Lane peripheral	310	3.1	**Crossbar**	3502 μm^2	1.79 mW
Output register	1646	634	**Cluster**	33042 μm^2	15.9 mW
Input register	2514	1011	**Tile**	0.264 mm^2	127.6 mW
Drive register (×2)	143	2.1	**Chip**	**136 mm^2**	**65 W**

We considered 128 × 128 crossbars that, similar to the other PIM designs, allows sharing an ADC to sense and digitize the current of all 128 bitlines within the 100 ns read latency of the rows in a time-multiplexed manner [133, 51]. Table 3.5 summarizes the PIM parameters. For the 128×128 array, sample-and-hold (S&H), and router, we used the same parameters of [51]. We characterized the digital components , i.e., DAC buffers, lane peripheral (8-bit column registers, tree-adder, and shift-and-and unit), hash and decoder unit, by implementing them in Verilog and synthesizing using a 14 nm standard cell library using Synopsis Design Compiler. We implemented the Murmur3 [6] hash as a three-stage pipeline. For the input and output register of the clusters, we used Artisan Memory Compiler using the same process node. For the ADC, we considered the 8-bit ADC fabricated in [87] and scaled the parameters to 14 nm according to [142] to match the process technology of the other peripherals. We consider a total PIM capacity of 512 Mbit, arranged as eight crossbars per cluster, and eight clusters per tile, making total 32,768 crossbars (512 tiles) while keeping the area and power consumption reasonable. Unlike [133], we consider an 8-bit column register (C-REG) for each bitline, which is imperative to latch the data before adding up all the bitlines' results. In addition, in case of bundling, each bitline is independent. The column registers are required to hold the data during writing them to the cluster output register using the shared bus. As a result, area of the lanes contribute to 60% of the total area. The total CMOS circuity, including the lanes, hash, decoder, registers and routers contribute to 75% of the area and 12% of the total power. The ADCs consume 73% of the total power. It is noteworthy that considerable research has focused on using less-precision ADC for PIM. In particular, due to its error resiliency, HDC has shown virtually no accuracy loss by replacing the 8-bit ADCs with 4-bit, which can reduce the ADC overhead exponentially [95].

Performance: Table 3.4 summarizes the performance details of implementing the proposed encoding methods in PIM. We developed a cycle-accurate simulator using Python that emulates the encoding and learning functionality and estimates performance. The "Allocated crossbars" column indicates the number of crossbars of the embedding (M) matrix for numeric encoding, and level vectors for categorical encoding, per one input. Multiple inputs are being processed in parallel. "Utilization rate" shows the percentage of active crossbar rows. "Encoding cycles" shows the number of memory cycles (100 ns each) for each encoding approach. The "Throughput" column reports the throughput in terms of million inputs that can be encoded per second using all crossbars of the PIM chip. The PIM architecture contains 32,768 crossbars, and can simultaneously process large number of inputs, leading to massive throughput.

The `OR` and `SUM` only differ in quantizing the categorical encoding, which is carried out after the encoding. Thus, these two encodings exhibit the same crossbar usage and latency. Notice that the number of allocated crossbars for the categorical encoding in `OR` and `SUM` encodings is higher than the `No-Count` (40 versus 20) as in the former ones the numeric encoding takes 81 cycles; hence, to keep up with the performance of numeric encoding, more crossbars are designated for the categorical encoding, at the cost of lower utilization rate. The numeric and categorical encoding are carried out concurrently. `No-Count` only performs categorical encoding which needs significantly less resources per input and achieves higher throughput by better input-level parallelism. `No-Count` assigns minimum number of crossbars to store the level vectors to improve the utilization rate and maximize the throughput.

Comparison with CPU

Encoding: Fig. 3.7(a) shows the encoding throughput in terms of number of inputs encoded in a second. We performed the CPU experiments using a system with Intel Core i7-8700K 3.70 GHz CPU with 64 GB memory. Notice that performance of encoding step is independent of the subsequent quantization, combining, and learning steps. The bars labeled as `No-Count` only consider the categorical data. The PIM results are the same reported in Table 3.4. When considering both numeric and categorical data, FPGA and PIM achieve 81$\times$ and 1177$\times$ speedup over CPU, respectively. Without the numeric data (i.e., `No-Count` setting), which makes the CPU encoding relatively faster, FPGA and PIM yield 11$\times$ and 414$\times$ speedup over CPU, respectively.

To account the power consumption differences, Fig. 3.7(b) compares the throughput per Watt (input per Joule). From previous subsections, FPGA and PIM consume a power of 30 W and 65 W, respectively. We estimated the CPU power consumption using CPU energy meter tool [99], and observed an encoding power of 88 W. Accordingly, throughput/Watt of FPGA (PIM) over CPU is 246× (1594×) when considering both categorical and numeric data, and 33× (560×) in `No-Count` case.

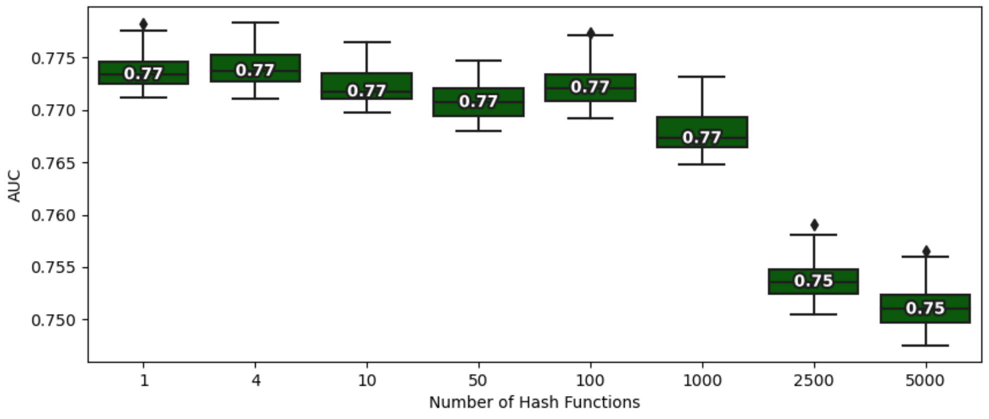
3.8 Conclusion

This chapter explores methods based on hashing as a means to alleviate the burden of materializing a large "embedding" matrix that is used to map symbols from their ambient representations into HD-space. We introduce a formal model that allows one to compare different encoding functions based on their ability to substantiate linear separators in HD space. We use this model to compare architectures for encoding high-cardinality categorical data, with more traditional approaches that bundle together a set of "codewords" which are instantiated by sampling from some distribution over the HD-space and stored in a large codebook. We show that both methods are able to achieve separability in HD-space when the input data can be modeled as a sequence of sets. In this case, linear separators in HD space approximate linear separators fit on the characteristic vectors representing the input data. However, hashing based methods are able to achieve this in a far more efficient way, by only storing a small number of hash-functions and constructing the encodings on-the-fly. We provide novel analysis for this setting that bounds the encoding-dimension and number of hash-functions that are sufficient to achieve separability in HD-space. Implementation in hardware confirms that the hashing-based approaches studied in this chapter can offer substantial performance improvements when compared to other approaches for codeword "re-materialization" that have been proposed in the literature. A seeming limitation of the approach discussed in this chapter is that we require the data to be separable to begin with. However, in the subsequent chapter we shall see that the notion of a dot-product-preserving encoding is, in fact, very flexible and can be adapted to handle a much wider range of situations in the input data.

Chapter 3 contains material from "An Analysis of Hashing Architectures for Scalable

Hyperdimensional Computing," by Anthony Thomas, Behnam Khaleghi, Tianqi Zhang, Weihong Xu, Gopi Krishna Jha, Nageen Himayat, Ravi Iyer, Nilesh Jain, Tajana Rosing, as it was submitted to IEEE Transactions on Neural Networks and Learning Systems. The dissertation author was the primary investigator and author of this paper.

(A) Effect of Number of hash-functions on Model Performance



(B) Effect of Encoding Dimension on Model Performance

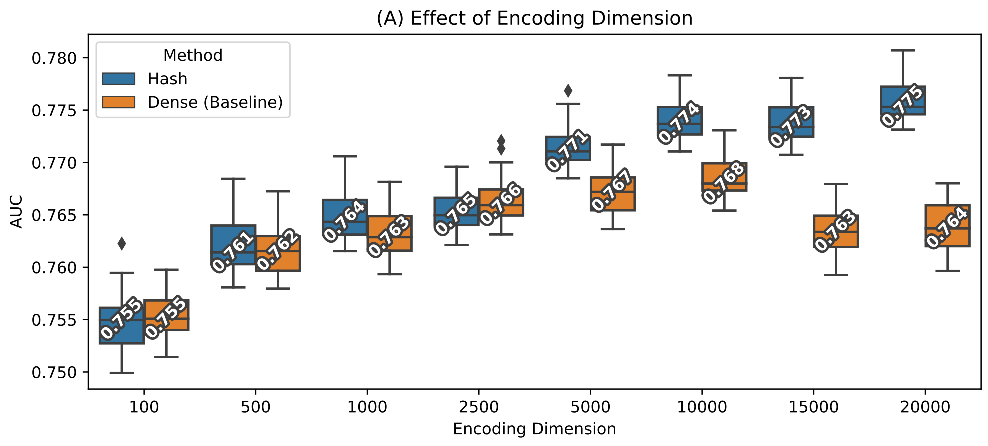


Figure 3.5. Evaluating the impact of categorical encoding dimension and number of hash-functions on model performance. Box plots show the distribution of AUC on non-overlapping groups of $100,000$ samples. The shaded box indicates the 1-st and 3-rd quartile. The solid line indicates the median and the whisker length is $1.5\times$ the IQR. The bundling method is concatenation, the numeric encoding method is a dense random projection ($d = 10,000$), and $k = 4$.

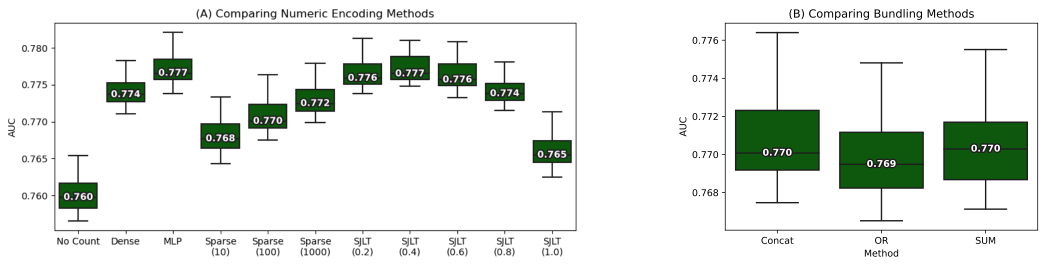


Figure 3.6. Panel (A) compares methods for encoding numeric data. “Dense” indicates the baseline signed random-projection described in Equation 3.3. “Sparse (k)” is the sparse random-projection scheme described in Equation 3.4, where k is the number of non-zero coordinates in the output. “SJLT (p)” indicates the SJLT scheme described in Equation 3.5, where p is the probability that a coordinate in the embedding matrix is non-zero. SJLT encodings are quantized using the sign function. “MLP” is a simple neural network model, and “No-Count” omits numeric data entirely. Panel (B) compares methods for bundling the numeric and categorical data as described in Section 3.5. Box plots are as described in Figure 3.5.

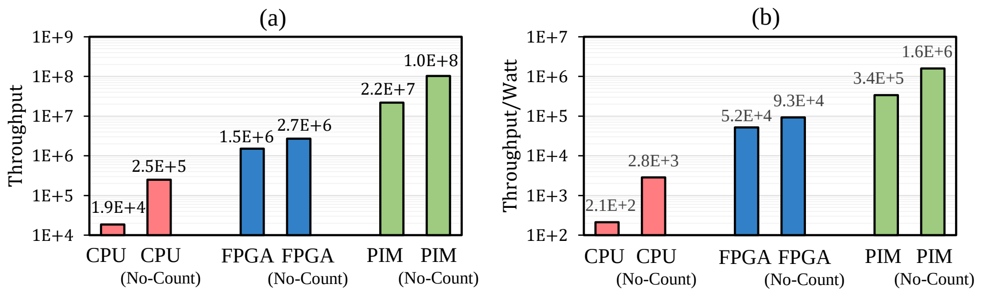


Figure 3.7. (a) Throughput (inputs per second) and (b) Throughput/Watt comparison of the encoding step for different platforms. The `No-Count` encoding omits the numeric data.

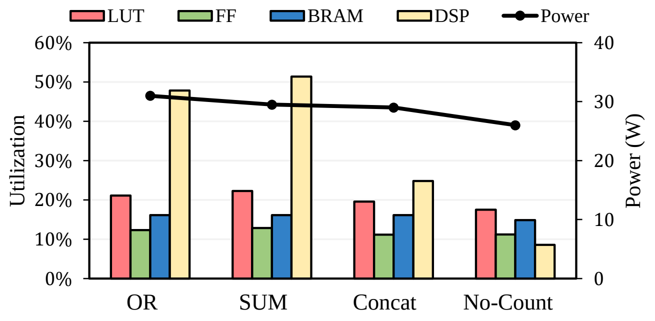


Figure 3.8. Resource and power utilization for different combining methods ($d = 10{,}000$). The Alveo U280 device contains 1157K Look-Up Tables (LUT), 2384K Flip-Flops (FF), 2016 BRAMs, and 9024 DSPs.

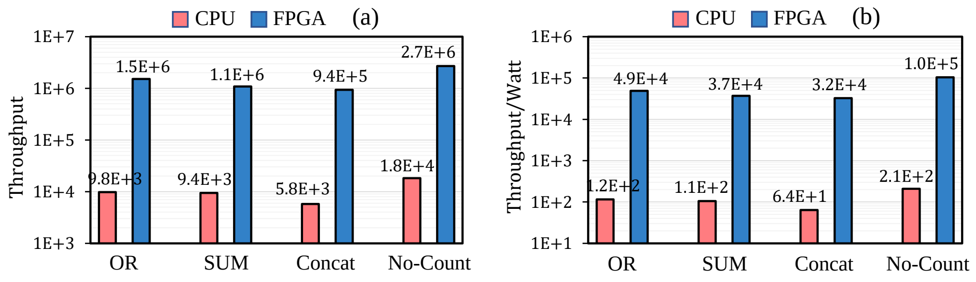


Figure 3.9. End-to-end (a) throughput (inputs per second) and (b) throughput/Watt comparison of FPGA and CPU implementation.

Chapter 4

Statistical Learning with HDC

In the following chapter, we significantly generalize the results of Chapters 2 and 3 concerning learning from HD representations to cover a wider range of learning problems, and to relax the assumption that perfect separability be achievable in HD space, which may be an unrealistic ask in practice. To do so, we build on the idea of a dot-product-preserving encoding that was used in the previous chapter. The key idea of the formal model considered there was that linear separability is definitely achievable in HD space, for a suitably large encoding dimension, if the encodings preserve dot-products with respect to the ambient representation of the data, and the data is separable in that representation with a positive margin. The assumption that the data is separable in its ambient representation is restrictive, but a simple observation is that we can always compose the HD-encoding ϕ with some other embedding ψ. Thus, if we have some embedding ψ that represents the data in an inner-product space in which it is separable, we can simply apply the machinery of the previous chapter on $\psi(x)$. Moreover, Theorem 21 does not actually require us to materialize $\psi(x)$. It is enough that ψ exist, and that $\langle \phi(x), \phi(x') \rangle \approx \langle \psi(x), \psi(x') \rangle$. That is to say, we can reap the benefits of ψ without incurring the cost of actually computing it. In the machine learning literature, similarity functions that are induced by inner-products between embeddings of data are typically known as "kernel functions," and are at the center of an enormously successful and diverse family of techniques called kernel methods, that, much like HDC, rely on unique properties that are elicited from high-dimensional embeddings of data [135, 151].

The basic premise of this chapter is that many of the encoding functions encountered in practice can be interpreted as preserving various kinds of kernels on the input data, and that this choice of underlying kernel plays a fundamental role in determining the capabilities of linear functions in HD space. Understanding such functions is important because the most commonly used approaches to learning in HDC amount to fitting linear functions in HD space. Moreover, in a learning setting, one is typically interested in developing a model that can be used to *predict* some quantity of interest about new data that has not been seen before, and a natural question of interest is how the choice of encoding function, dimension, and precision, effect the ability to fit useful models from HD representations of data. To address this question formally, we study learning with HDC using techniques from the literature on statistical learning theory, which allow us to provide a more precise characterization of this question than is available in prior work in the literature on HDC. We summarize the key results of this chapter as follows:

(1) For a popular approach to learning based on fitting linear functions to encoded representations of data, the space of learnable models is completely determined as the set of all possible *linear* combinations of encodings of data. For an important family of encoding functions that can be interpreted as approximating some underlying kernel of interest, fitting linear functions in HD space approximates kernel machines fit on the underlying data. A significant portion of the chapter will be devoted to quantifying this statement and making it rigorous. From a practical perspective, this insight is useful both for interpreting HD models, and because there is a very large theoretical and applied literature on learning with kernels, much of which can be imported into the HD setting.

(2) We extend the classic capacity theory of HDC [105, 48, 29] to address learning settings, and develop a model, using techniques from statistical learning theory, that allows us to formally analyze the relationship between encoding dimension, precision, the number of samples presented to the learning algorithm, and performance on a particular task.

(3) Using this framework, we show that, for a broad family of encoding functions and learning problems expressed on HD representations, the loss of a model learned from d-dimensional encodings, and represented using b-bits of precision per-coordinate converges to the best achievable risk, in expectation over randomness in the encoding and given perfect knowledge of the data distribution, at a rate $O(\frac{1}{\sqrt{d}}(1+\frac{1}{2^b-1})+\frac{1}{\sqrt{n}})$, up to terms that depend on the particulars of the HD architecture and learning problem in question.

4.1 Background and Related Work

In the following section we review some salient background on kernel methods and statistical learning theory.

4.1.1 Background on Kernel Methods

We here give a brief introduction to the essential terminology of kernel methods. The following definitions can be found in [151]. Let $\mathscr{X}$ be a non-empty set. We call a function $k : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ a kernel if and only if it is symmetric and positive-semi-definite (PSD).[1] For any kernel, there exists a Hilbert space[2] $\mathscr{H}$, and a map $\psi : \mathscr{X} \to \mathscr{H}$, called the "feature map," such that $k(x,x') = \langle \psi(x), \psi(x') \rangle$. Typically, one views k as a type of similarity function on $\mathscr{X}$, defined by a two-stage operation in which one embeds data into $\mathscr{H}$ under $\psi(x)$, wherein similarities are measured using inner-products. However, for many kernels of practical interest, the kernel function can be evaluated directly on the low-dimensional representations of the data, and the embedding need not be materialized explicitly. The polynomial kernel $k(x,x') = (1+\langle x,x' \rangle)^p$ is a canonical such example. The Gaussian kernel $k(x,x') \propto \exp(-\|x-x'\|_2^2/2)$ is another. This feature is of interest since it allows one to work with feature maps of very high, or even infinite, dimension.

[1]Recall that a function $k : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ is PSD if, for all n and all sets $\{x_1, ..., x_n\} \subset \mathscr{X}^n$, the matrix $K_{ij} = k(x_i, x_j)$ is positive semi-definite.

[2]A Hilbert space is a complete inner-product space

Any kernel induces a space of functions that can be evaluated pointwise via inner-products in $\mathscr{H}$. That is, viewing some $f \in \mathscr{H}$ as a function, we evaluate it at x via $f(x) = \langle \psi(x), f \rangle$. The space of all such functions (generated by a particular kernel) can be characterized as space of all possible linear combinations of feature-space representations of data. That is:

$$\mathscr{F} = \left\{ \sum_{i=1}^{n} \alpha_i \psi(x_i) \,\middle|\, n \in \mathbb{Z}^+, \alpha_i \in \mathbb{R}, \sum_{ij} \alpha_i \alpha_j k(x_i, x_j) < \infty \right\}.$$

The space $\mathscr{F}$ is called the "reproducing Kernel Hilbert space" (RKHS) of k. In general, the feature map of a kernel is not unique, but the RKHS is. That is, there may be many ways to embed data that yield the same kernel, but all such embeddings define the same RKHS.

Generally speaking, kernel methods extend classic simple learning algorithms like SVMs, Perceptrons, K-means clustering, PCA, and least-squares regression, to non-Euclidean settings by running these procedures on the embeddings of data, and can usually be interpreted as search procedures over functions in an RKHS [135, 134]. That is to say, a "kernel SVM" is simply a standard linear SVM run in the feature space of a kernel, and returns a function from the corresponding RKHS. Per the above, this function can be interpreted as a weighted linear combination of feature-space embeddings of data points. However, in algorithmic realizations of kernel methods, the embedding is typically implicit and inner products are evaluated by the kernel function directly.

We remark that the feature map of kernel methods serves almost the same intuitive function as encoding in HDC. That is, to represent the data in such a way that simple notions of similarity between encodings capture some salient, and possible more complex, notion of similarity on the raw data. However, in kernel methods, the feature map is often only defined implicitly, and may not satisfy other desiderata of HD computing. For instance, in the feature-maps of kernels, the coordinates may have specific meaning which violates the "distributed" paradigm advocated in HDC, and are usually of high-precision. However, there exist kernels for which both of these desiderata can be satisfied [119].

4.1.2 Statistical Learning Theory

In a learning setting, one typically wishes to use data to build a predictive model that can be used to infer information about some outcome of interest. There are a number of formalisms available in which one may study this problem rigorously. A particularly compelling and influential one is statistical in nature: one views the data as being generated by some underlying, and typically unknown, probability distribution. The goal of learning is to develop a predictive model whose error under this distribution is small [149, 134, 151]. This is typically formalized using the notion of risk. Let $\ell : \mathbb{R}^2 \to \mathbb{R}^+$ be a non-negative loss function used to measure the quality of a prediction, let P be a probability distribution on $\mathscr{X} \times \mathscr{Y}$, and let $\mathscr{F} : \mathscr{X} \to \mathbb{R}$ be a class of functions, sometimes also called the "hypothesis class." The risk of $f \in \mathscr{F}$ is defined to be $\mathbb{E}_{(x,y)\sim P}[\ell(f(x),y)]$. That is, the risk is the expectation of the loss for a random example drawn from P. We will omit the $(x,y) \sim P$ subscript where it is clear from context what the expectation is taken with respect to. We briefly remind the reader that ℓ is said to be ρ-Lipschitz if $|\ell(t,y) - \ell(t',y)| \leq \rho|t-t'|$, for all $t,t' \in \mathbb{R}$.

The challenge of this model is that we do not know P, and so cannot actually compute the risk. Instead, we must content ourselves to infer something about it based on observed data, which we assume to be generated by sampling from P. The general idea is to select a candidate hypothesis $\hat{f} \in \mathscr{F}$ on the basis of data, and our hope is that $\hat{f}$ achieves low-risk with respect to P. There are then two primary questions of interest: first, by how much does the empirical risk of $\hat{f}$, computed from a set of samples from P, underestimate its true risk? That is, given a set of n samples $\{(x_1,y_1),...,(x_n,y_n)\}$ drawn i.i.d. from P, how big can:

$$\mathbb{E}[\ell(\hat{f}(x),y)] - \frac{1}{n}\sum_{i=1}^{n}\ell(\hat{f}(x_i),y_i),$$

be? And second, how close is the risk of $\hat{f}$ to the best achievable risk over hypotheses in $\mathscr{F}$?

That is, how big can:

$$\mathbb{E}[\ell(\hat{f}(x),y)] - \inf_{f \in \mathscr{F}} \mathbb{E}[\ell(f(x),y)],$$

be? One of the key goals of statistical learning theory is to obtain bounds on these quantities.

A key ingredient for obtaining such bounds is to quantify the "richness" of the hypothesis class. The basic idea is that if the hypothesis class is extremely rich (e.g. able to fit almost any relationship between input and output), then we run a greater risk of mining fictitious relationships in the data. This situation is usually called "overfitting." We here use the following notion to quantify the richness of the hypothesis class [134]:

Definition 7. ***Rademacher Complexity****: Let $\mathscr{D} = \{x_1, ..., x_n\} \subset \mathscr{X}^n$ be a set of samples drawn i.i.d. from a distribution P on $\mathscr{X}$, let $\mathscr{F}$ be a class of functions $\mathscr{X} \to \mathbb{R}$, and let $\sigma = (\sigma_1, ..., \sigma_n)$ be independent random variables where $\sigma_i \sim$ Unif$(\{+1, -1\})$. The empirical Rademacher complexity of $\mathscr{F}$, with respect to $\mathscr{D}$ is:*

$$\mathfrak{R}(\mathscr{F} \circ \mathscr{D}) = \mathbb{E}_\sigma \left[\sup_{f \in \mathscr{F}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i f(x_i) \right].$$

Intuitively, the Rademacher complexity measures the richness of a function class in terms of its ability to fit a completely random sequence of binary labels. A function class capable of fitting a completely random string of labels is dangerously rich (e.g. can easily overfit), and one must exercise caution when trusting its predictions. The Rademacher complexity can be used to obtain bounds on the excess risk [10]. We use the following result, which can be found in [134, Theorem 26.5]:

Theorem 27. *Let $\mathscr{D} = \{(x_1, y_1), ..., (x_n, y_n)\}$ be a set of samples drawn i.i.d. from a distribution P on $\mathscr{X} \times \mathscr{Y}$. Let $\mathscr{F} : \mathscr{X} \to \mathbb{R}$ be a class of functions, let ℓ be a ρ-Lipschitz loss function satisfying $|\ell(f(x),y)| \leq c$, for all x, y. Then with probability at least $1 - \delta$ over samples of size n,* *for all $f \in \mathscr{F}$:*

$$\mathbb{E}[\ell(f(x),y)] - \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i),y_i) \leq 2\rho\mathfrak{R}(\mathscr{F} \circ \mathscr{D}) + 4c\sqrt{\frac{2\ln(4/\delta)}{n}}.$$

The result says that the maximum discrepancy between the empirical risk, estimated from a set of n samples, and the true risk can be upper-bounded in terms of the Rademacher complexity of the underlying function class, the Lipschitz constant (e.g. smoothness) of the loss, and its absolute bound, where smaller values of all three quantities means that empirical risk converges to its population analogue faster (in n). One might then hope to restrict attention to function classes with small Rademacher complexity. However, an overly restrictive function class may be unable to offer sufficient flexibility to capture *real* relationships in the data and lead to poor performance. The "art" of machine learning is to design the hypothesis class in such a way that it is just rich enough to model the process of interest. This is, of course, a complex problem, but there are a variety of techniques that can be used to adaptively control the complexity of the hypothesis class [151].

As a corollary of the previous Theorem, we obtain the following useful result, which bounds the maximum possible discrepancy between the hypothesis that attains minimum empirical risk with respect to a particular sample from P, and the best achievable risk over all possible hypotheses [134, Theorem 26.5]:

Theorem 28. *In the context of Theorem 27, let:*

$$\hat{f}_n = \operatorname*{argmin}_{f \in \mathscr{F}} \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i),y_i), \textit{ and } f^* = \operatorname*{argmin}_{f \in \mathscr{F}} \mathbb{E}_{(x,y)\sim P}[\ell(f(x),y)].$$

Assuming the latter quantity exists, with probability at least $1-\delta$ over samples of size n:

$$L_P(\hat{f}_n) - L_P(f^*) \leq 2\rho\mathfrak{R}(\mathscr{F} \circ \mathscr{D}) + 5c\sqrt{\frac{2\ln(8/\delta)}{n}}.$$

We will make repeated use of this equation, which is cumbersome to write out, so let us introduce the following notation. For any specific sample $\{(x_1,y_1),...,(x_n,y_n)\}$ of size n drawn from a distribution P, and class of functions $\mathscr{F} : \mathscr{X} \to \mathbb{R}$, let us define:

$$\hat{f}_n = \underset{f\in\mathscr{F}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i),y_i), \text{ and } f^* = \underset{f\in\mathscr{F}}{\operatorname{argmin}} \mathbb{E}_{(x,y)\sim P}[\ell(f(x),y)],$$

assuming the later quantity exists. Finally, for any $f \in \mathscr{F}$, let us denote by:

$$L_n(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i),y_i), \text{ and } L_P(f) = \mathbb{E}_{(x,y)\sim P}[\ell(f(x),y)].$$

4.2 Encoding and Kernel Approximation

Recall that, in HDC, data is mapped from its ambient representation $x \in \mathscr{X}$, to some inner-product-space $\mathscr{H} \subset \mathbb{R}^d$ under an encoding function $\phi : \mathscr{X} \to \mathscr{H}$. It is mechanically the case that every encoding function induces a kernel on $\mathscr{X}$, via a dot-product in $\mathscr{H}$: $k_\phi(x,y) = \langle \phi(x), \phi(y) \rangle$. As noted in previous chapters, one typically tries to design ϕ so that $k_\phi(x,y) \approx k(x,y)$, where k is some underlying kernel of interest on $\mathscr{X}$. We briefly reiterate some examples from the previous chapters.

Example 1. ***Bundles of Bindings***. *Let $\mathscr{F}$ be a set of n features, and $\mathscr{A}$ be a set of m values. To each $f \in \mathscr{F}$ and $a \in \mathscr{A}$, we assign "codewords" $\psi(f), \phi(a) \sim Unif(\{\pm 1\}^d)$. To represent a pair (f,a) we bind together their embeddings using element-wise multiplication: $\psi(f) \otimes \phi(a)$. To represent a collection of such pairs $x = \{(f_i,a_i)\}_{i=1}^m$, we bundle together the bindings: $\phi(x) = \bigoplus_i \psi(f_i) \otimes \phi(x_i)$. When the bundling operator is linear, the induced kernel is a form of set-intersection: $k(x,x') = \mathbb{E}[\langle \phi(x), \phi(x') \rangle] = d|x \cap x'|$, where $x \cap x'$ is the number of (f,a) pairs on which x,x' agree (e.g. Theorems 12, 22).*

Example 2. ***Sparse Binary Codes***. *A variant of the above arises when $\phi(a)$ is s-sparse and binary, and $\oplus$ is the logical-or [84, 29]. Then, for some $x \subset \mathscr{A}$, $\phi(x) = \bigoplus_{a\in x} \phi(a)$ is a form of*

Bloom filter [14], and $\mathbb{E}[\langle \phi(x), \phi(x') \rangle]$ again measures the set intersection (e.g. Theorem 24).

Example 3. ***Random Projection***. *Let $\mathscr{X} \subset \mathscr{S}^{m-1}$ (the m-dimensional unit-sphere). To encode a point $x \in \mathscr{X}$, let us draw $w_1, ..., w_d \sim$ Unif$(\mathscr{S}^{m-1})$, and define $\phi(x)_i = (\langle w_i, x \rangle)/\sqrt{d}$, for $i = 1, ..., d$. Then $k(x,x') = \mathbb{E}[\langle \phi(x), \phi(x') \rangle] = \langle x, x' \rangle / m$, which is the linear kernel (scaled by the dimension of the underlying data). A slightly more conventional approach would be to eliminate the scaling with m by choosing $w_i \sim \mathscr{N}(0, I_m)$. A similar result holds for choosing $w_i \sim$ Unif$(\{\pm 1/\sqrt{m}\}^m)$ [2]. The quantized variant $\phi(x)_i = \text{sign}(\langle w_i, x \rangle)/\sqrt{d}$ induces the kernel $k(x,x') = \mathbb{E}[\langle \phi(x), \phi(x') \rangle] = 1 - 2\cos^{-1}(\langle x, x' \rangle)/\pi$ (e.g. Theorem 15), which is the angular kernel [61]. This technique also arises commonly in the literature on locality sensitive hashing [5].*

Example 4. ***Random Fourier Features***. *The previous method can be generalized to capture other notions of similarity. Let $w_1, ..., w_d \sim \mathscr{N}(0,1)$, and define $\phi(x)_i = \sqrt{2}\cos(\langle w_i, x \rangle + b_i)$, where $b_i \sim$ Unif$([0, 2\pi])$. Then, it can be shown that $\mathbb{E}[\langle \phi(x), \phi(x') \rangle] \propto \exp(-\|x - x'\|_2^2)$, which is the Gaussian kernel [119]. In fact, this technique, called "random Fourier features" (RFF) is much more general and can be adapted to capture any shift-invariant kernel on $\mathbb{R}^m$ of the form: $k(x,x') = k(x - x')$. Closely related approaches arise in the HDC literature under the names "fractional power encoding" [47], "nonlinear encoding" [159], and "spatial semantic pointers" [86]. There are a variety of techniques for producing low-precision random-Fourier features: [113] uses a thresholding scheme to produce binary approximations, and [156, 157] use randomized rounding to produce approximations of arbitrary precision.*

To unify these approaches into a common framework, we introduce the following definition:

Definition 8. ***Kernel Preserving Encoding***: *We say ϕ is $\Delta(d,n)$-preserving for a kernel k if, for any set of n points $\{x_1, ..., x_n\}$, it is the case that:*

$$k(x,x') - \Delta(d,n) \leq \langle \phi(x), \phi(x') \rangle \leq k(x,x') + \Delta(d,n). \tag{4.1}$$

Definition 8 can be seen as a generalization of the notion of dot-product-preserving encoding from the previous chapter (i.e. Definition 6) which applies to any Hilbert space embedding of x. That is, the property can be interpreted as in Definition 6, but with respect to dot-products on feature-space embeddings of data: $\psi(x)$. The examples described above can all be characterized by designing a random-map $\varphi : \mathscr{X} \to \mathbb{R}$ such that $\mathbb{E}[\varphi(x)\varphi(x')] = k(x,x')$, and then setting $\phi(x) = (\varphi_1(x),...,\varphi_d(x))/\sqrt{d}$ (or $\sqrt{s}$ for the sparse method), where $\varphi_1,...,\varphi_d$ are independent instantiations of φ. Throughout this work, we will always assume that ϕ is bounded in the sense that $|\phi(x)_i| \leq \sqrt{M}$ for all x and i. In this case, we can give the following general bound on $\Delta(d,n)$:

Theorem 29. *Let $\varphi : \mathscr{X} \to \mathbb{R}$ be a random map satisfying $|\varphi(x)| \leq \sqrt{M}$ for all $x \in \mathscr{X}$, and $\mathbb{E}[\varphi(x)\varphi(x')] = k(x,x')$, for some positive-definite kernel k. Then, for any set of n points $\{x_1,...,x_n\}$, the map $\phi : \mathscr{X} \to \mathscr{H}$ defined element-wise by $\phi(x)_i = \varphi_i(x)/\sqrt{d}$, is $\Delta(d,n)$-preserving with probability at least $1-\delta$, where:*

$$\Delta(d,n) \leq \sqrt{\frac{4M^2}{d} \ln \frac{n}{\delta}}.$$

The proof is a direct application of Hoeffding's inequality and the union bound. The result implies that, to preserve pairwise similarities (as defined by the kernel k) between any set of n points to error ε, it is sufficient to use $d = O((M/\varepsilon)^2 \log n)$ dimensions. A further consequence of the this result is that $\|\phi(x)\|_2$ is concentrated around $\sqrt{k(x,x)}$. To simplify discussion, we will assume that $|k(x,y)| \leq 1$ with equality if and only if $x = y$, whereupon $L = \max_x \|\phi(x)\|_2 = O(1)$. Neither of these assumptions are crucial, and our results can be generalized in a relatively straightforward way to handle relaxations of them.

4.2.1 Related Work

While there is a large empirical literature on learning with HDC [114, 85, 79, 116, 18, 82], the theoretical literature has traditionally focused primarily on analyzing their storage capacity

[106, 48, 144, 29]. That is, on addressing how particular data items (or collections thereof) can be represented in a manner that permits reliable recovery. The question of learning from HD representations, that is using them to *predict* some outcome of interest, is less well studied from a formal perspective. In general, the fact that encoding can approximate certain types of kernels is well known [106, 144, 47, 155]. However, the implications of this for learning have not been thoroughly explored.

Recent work in [155] showed the existence of kernels that cannot be represented exactly by certain HD encoding procedures, and that this implies the existence of data distributions for which certain learning algorithms will yield sub-optimal results. However, the existence of distributions for which a particular encoding will fail to produce optimal results does not seem generally informative for understanding its utility in learning applications. Work in, [47] studied how bundling and binding could be interpreted as manipulating functions, and observed that certain learning algorithms expressed on HD representations could be interpreted as kernel machines, but their work was focused on a specific encoding architecture and does not provide any analysis in the finite-dimensional setting one is limited to in practice. We are aware of no work in the HDC community that rigorously studies how the choice of encoding dimension and precision effect the performance of learning from HD representations, nor are we aware of work studying the statistical aspects of learning with HDC (e.g. sample complexity). The closest to this is [144, 145] (e.g. Chapters 2 and 3 of this work), who provide conditions under which algorithms based on finding separating hyperplanes in HD space will succeed, but require the data to be separable to begin with, which is restrictive in practice. Our work provides the first formal treatment of these issues in significant generality.

Finally, there is a large and active body of work on learning from random-distributed representations of data in the broader machine learning community under the name "random Fourier features" (RFF) [119, 120, 156, 139]. This line of work is focused on a specific method for generating embeddings that approximate shift-invariant kernels, but many of the basic insights and analytic techniques of this line of work hold for a more general class of encoding techniques that can be interpreted as random approximations to kernels. One of our goals is to clarify the relationship between these closely related areas of work.

4.3 Learning with HDC

The remainder of the paper is devoted to analyzing the use of VSA representations in learning algorithms. The classical notion of storage capacity for VSAs quantifies the relationship between the encoding dimension, the number of data points stored in an HD representation, and the probability of answering membership queries (e.g. "decoding") correctly. In a learning setting, the natural analogue is the error, as measured by some non-negative loss function $\ell : \mathbb{R}^2 \to \mathbb{R}^+$, between the prediction generated by a model $f(x)$ obtained from the HD representations of points, and the ground truth y. Following the statistical model in the previous section, we assume the data is generated by some unknown distribution P on $\mathscr{X} \times \mathscr{Y}$, and our goal is to gain control over the risk of models fit using the HD representations of data, which is defined as the expected loss under P. That is: $\mathbb{E}_{(x,y)\sim P}[\ell(f(x), y)]$. This is the notion we here adopt to quantify learning "capacity." As a concrete example, in a classification setting, one might take $\ell(f(x), y) = \mathbb{1}(\text{sign}(f(x)) \neq y)$, in which case the risk is the probability the classifier makes a mistake. Our goal is to study how the choice of encoding function, dimension, precision, and the number of samples presented to a learning algorithm effects the risk.

To make progress on this question, we must impose some restriction on how the encodings are used. In general, the encodings are merely vector representations of data and could be used as input to any algorithm capable of ingesting such data. In practice, many of the approaches used for learning in HDC can be characterized as fitting linear functions in HD space [114, 62, 85, 47]. Such methods are amenable to hardware implementation [114, 62], can often be updated incrementally as new data is received, and are also important in neuroscience: many canonical neural learning rules like Hebb and Oja's rules amount to learning linear functions in the data-space [59]. In particular, we are here interested in methods that make predictions using functions of the form $f(x) = \langle \phi(x), \theta \rangle$, where $\theta \in \mathscr{H}$, and ϕ is the HD encoding. We call the triplet (ϕ, θ, ℓ) the "HD model."

To motivate our focus on this approach to learning and make our discussion more concrete, let us revisit the scheme for classification introduced in Chapter 1. Let $\{(x_i, y_i)\}_{i=1}^n$ be a set of labeled data, where $x_i \in \mathscr{X}$ is an input and $y_i \in \{1, ..., c\}$ is a categorical output. We construct a representer for each class as a weighted bundle of the training data:

$$\theta_j = \bigoplus_{i=1}^{n} \alpha_{ij} \phi(x_i), \tag{4.2}$$

where $\alpha_{ij} \in \mathbb{R}$ is a weight assigned to the i-th example for the j-th class. As discussed previously, notable exemplars of this scheme are to simply bundle together to training data corresponding to a particular class (see: [85, 116] among many others), in which case $\alpha_{ij} = \mathbb{1}(y_i = c_j)$, and to apply the Perceptron algorithm [62, 65, 95], in which case $\alpha_{ij} \in \{+1, 0, -1\}$, after the first pass over the training data, depending on how the algorithm made a mistake on the i-th example. One then predicts a label for a query point x_0 according to:

$$\hat{y}(x_0) = \underset{j \in [c]}{\operatorname{argmax}} \langle \phi(x_0), \theta_j \rangle. \tag{4.3}$$

In general, we may interpret this as associating each class with a function $f_j : \mathscr{X} \to \mathbb{R}$, that is linear in $\mathscr{H}$ (whether or not θ is a linear function of the $\phi(x_i)$'s), and evaluated pointwise via $f_j(x) = \langle \phi(x), \theta_j \rangle$, which is a common approach to multi-class learning [88]. The dot-product is sometimes replaced with the cosine or Hamming similarity, but both of these can be treated as special cases of the above. That is to say, they can still be interpreted as linear parametric functions in HD space.

Of course, these are merely two examples of a much more general paradigm in which one constructs a vector of parameters for a linear function in $\mathscr{H}$ as a weighted superposition of encodings training data. In the Perceptron, the weights are obtained by minimizing the hinge-loss $\ell(\langle\theta,\phi(x)\rangle,y) = \max(1-y\langle\theta,\phi(x)\rangle,0)$ (or its multiclass variant [134]) using online gradient-descent with a conservative update rule, but this general principle can be applied to a much more diverse family of losses, giving rise to techniques like SVMs, logistic-regression, and ridge-regression. All of these techniques are associated with a particular loss that can be minimized using online gradient descent [58], and leads to a vector of parameters θ that can be computed incrementally and written as a weighted sum of encodings of data.

4.3.1 The Hypothesis Space of an HD Architecture

The first question is to characterize more precisely how the encoding function determines the class of models that can actually be learned from data, typically called the "hypothesis space" [134]. We say that f is admissible if $f(x) = \langle\phi(x),\theta\rangle$ is bounded. For any θ generating an admissible f, it must be the case that:

$$\theta \in \mathscr{S} \subseteq \left\{ \sum_{i=1}^{n} \alpha_i \phi(x_i) \,:\, n \in \mathbb{Z}^+, \alpha_i \in \mathbb{R}, \sum_{ij} \alpha_i \alpha_j \langle\phi(x_i),\phi(x_j)\rangle < \infty \right\},$$

where $\mathbb{Z}^+$ denotes positive integers. To see that this entails no loss of generality, we can decompose $\theta = \theta_{\mathscr{S}} + \theta_{\mathscr{S}^c}$, where $\theta_{\mathscr{S}} \in \mathscr{S}$ and $\theta_{\mathscr{S}^c}$ is in the orthogonal complement of $\mathscr{S}$. Then, $\langle\phi(x),\theta\rangle = \langle\phi(x),\theta_{\mathscr{S}}+\theta_{\mathscr{S}^c}\rangle = \langle\phi(x),\theta_{\mathscr{S}}\rangle$, and so the only part of θ that is detectable by $\phi(x)$ must be in $\mathscr{S}$. Thus, the entire space of functions that can be described by $f(x) = \langle\phi(x),\theta\rangle = \sum_i \alpha_i k_\phi(x,x_i)$, can be equivalently represented as: (1) linear combinations of encodings of data points or (2) weighted sums of kernel functions induced by the encoding. This is true even if θ is formed using a "nonlinear" bundling operator. We can therefore interpret the sum as a universal bundling operator in the sense that it is sufficient to represent the entire hypothesis space of a VSA whether or not its native bundling operator is the sum. Of course, one may impose additional constraints on ϕ,θ (e.g. that they be of low-precision), but this can only serve to restrict the hypothesis space. Some of our results will be stated in terms of $\|\alpha\|_1 = \sum_i |\alpha_i|$. We denote by $\mathscr{S}_A$, the space of admissible θ, generated by weighted sums of data embeddings where $\|\alpha\|_1 \leq A$.

We note as well that $\mathscr{S}$ can also be characterized as the RKHS of k_ϕ, the kernel induced by encoding. For encodings satisfying the constraints of Theorem 29, we can then characterize the hyposthesis class resulting from taking expectations over randomness in the encoding as being the RKHS of k (the kernel approximated by the encoding). From a practical perspective, this is significant because different encodings measure similarity in different ways (e.g. approximate different kernels), and lead to hypothesis spaces of different capabilities. We describe the hypothesis spaces associated with several of the encoding methods discussed in Section 4.2 in the following.

Bundles of Bindings

Let $\mathscr{F}$ be a set of n features, and $\mathscr{A}$ be a set of m values. To each $f \in \mathscr{F}$ and $a \in \mathscr{A}$, we assign "codewords" $\psi(f), \phi(a) \sim \text{Unif}(\{\pm 1\}^d)$. To represent a pair (f,a) we bind together their embeddings using element-wise multiplication: $\psi(f) \otimes \phi(a)$. To represent a collection of such pairs $x = \{(f_i, a_i)\}_{i=1}^n$, we bundle together the bindings:

$$\phi(x) = \bigoplus_i \psi(f_i) \otimes \phi(x_i).$$

As noted above, when $\bigoplus$ is linear (e.g. the element-wise sum), one can see that this approximates a form of set intersection. Let $\mathscr{I} = x \cap x'$ be the set of features on which x and x' agree. Then:

$$\frac{1}{d}\langle \phi(x), \phi(x') \rangle = \frac{1}{d} \sum_{(f,a)\in\mathscr{I}} \|\psi(f) \otimes \phi(a)\|_2^2 + \Delta,$$

where Δ is a noise term cased by chance correlation between the embeddings of the features (e.g. $\psi(f_i)$). Taking expectations over randomness in the encoding, one can see that $\mathbb{E}[\Delta] = 0$, and so the kernel approximated is $k(x,x') = \frac{1}{d}\mathbb{E}[\langle \phi(x), \phi(x') \rangle] = |x \cap x'|$ (i.e. Theorem 12).

Now, let $\mathscr{S} = \mathscr{A} \times \mathscr{F}$, whereupon we may view $x \subset \mathscr{S}$. Let $s(x) \in \{0,1\}^{nm}$ be the characteristic vector of x, that is the vector that whose positive indices encode the identities of the symbols in x. Then, we can equivalently represent $k(x,x') = \langle s(x), s(x') \rangle$, and so we can interpret the encoding procedure above as preserving the linear kernel on the characteristic vectors encoding sets drawn from the universe $\mathscr{A} \times \mathscr{F}$. Therefore, functions of the form:

$$\begin{aligned} f(x) &= \left\langle \phi(x), \sum_{i=1}^{n} \alpha_i \phi(x_i) \right\rangle = \sum_{i=1}^{n} \alpha_i \langle \phi(x), \phi(x_i) \rangle \\ &\approx \sum_{i=1}^{n} \alpha_i \langle s(x), s(x_i) \rangle = \langle s(x), \tilde{\theta} \rangle, \end{aligned}$$

can be interpreted as approximating linear functions on the characteristic vectors of sets from $\mathscr{A} \times \mathscr{F}$.

Linear Random Projection

Let $\mathscr{X} \subset \mathscr{S}^{m-1}$ (the m-dimensional unit-sphere). To encode a point $x \in \mathscr{X}$, let us draw $w_1, ..., w_d \sim \text{Unif}(\mathscr{S}^{m-1})$, and define $\phi(x)_i = \langle w_i, x \rangle$, for $i = 1, ..., d$. Then, for any $i \in [d]$:

$$\mathbb{E}[\phi(x)_i \phi(x')_i] = x^T \mathbb{E}[w_i^T w_i] x' = x^T \left(\frac{1}{m} I_m \right) x' = \frac{1}{m}(x^T x'),$$

where I_m is the $m \times m$ identity matrix. Therefore, this encoding method preserves the linear kernel $k(x,x') = \langle x, x' \rangle$, scaled by a factor of $1/m$ (the dimension of the data). Thus, linear functions in ϕ-space approximate linear functions on $\mathscr{X}$.

Non-Linear Kernels

The previous encodings were simple to interpret because they induce linear kernels on the ambient representation of the data. However, this makes their hypothesis spaces somewhat limited, as they can only recover linear functions on the underlying data. However, the hypothesis spaces induced by some kernels can be significantly richer. For an important family of kernels, called "universal" kernels, the RKHS is dense in the space of all continuous functions on $\mathscr{X}$ [140].

To be slightly more formal, let $\mathscr{X} \subset \mathbb{R}^m$ be compact, let $C(\mathscr{X})$ be the space of all continuous functions on $\mathscr{X}$, and let k be a kernel on $\mathscr{X}$ with RKHS $\mathscr{F}$ as defined in Section 4.1.1. Then k is said to be universal, if for every $g \in C(\mathscr{X})$, and all $\varepsilon > 0$, there exists $f \in \mathscr{F}$, such that:

$$\sup_{x \in \mathscr{X}} |f(x) - g(x)| \leq \varepsilon.$$

That is to say, for every continuous function on $\mathscr{X}$, there is an RKHS function that arbitrarily well approximates it in a uniform sense.

The Gaussian kernel $k(x,x') \propto \exp(-\|x-x'\|_2^2/2)$ is a well known example of a universal kernel on $\mathscr{X} \subset \mathbb{R}^m$ [140, Example 1], and can be approximated using random Fourier features, by drawing $w_1,...,w_d \sim \mathscr{N}(0,1)$, and setting $\phi(x)_i = \sqrt{\frac{2}{d}}\cos(\langle w_i, x\rangle + b_i)$, where $b_i \sim \text{Unif}([0,2\pi])$.

The angular kernel $k(x,x') = 1 - 2\cos^{-1}(\langle x,x'\rangle)/\pi$ is universal on $\mathscr{X} \subseteq \mathscr{S}^{m-1}$ [61] (via [140, Corollary 10]), and can be approximated using the signed random-projection technique $\phi(x)_i = \text{sign}(\langle w_i, x\rangle)/\sqrt{d}$, where $w_i \sim \text{Unif}(\mathscr{S}^{m-1})$.

Fitting linear functions on these encodings can thus be substantially more powerful than linear functions on the ambient representation of the data. In fact, in expectation over randomness in the encoding, the hypothesis spaces of these encoding techniques can essentially recover the entire space of continuous functions on the input data. Of course, in practice, one is limited to finite-dimensional approximations, which introduces additional error, but the takeaway message is that certain encoding functions can induce rich hypothesis spaces that make linear functions learned in ϕ-space much more flexible than linear functions learned on the ambient representation of the data.

4.3.2 The Risk of the Hypothesis Class Induced by Encoding

In light of the above, we may refine our definition of risk to be the expected loss of functions in the hypothesis class induced by encoding: $\mathbb{E}_{(x,y)\sim P}[\ell(\langle\theta, \phi(x)\rangle, y)]$. Since one is limited to learning models in this class, the minimum risk over the hypothesis class may be non-zero. That is to say, there may exist no $\theta \in \mathscr{S}$ that is perfectly consistent with the underlying data. Moreover, the best achievable risk is, in general, an unknowable quantity since it is defined in terms of the underlying data distribution. In practice, one typically does have access to *samples* drawn from P (e.g. the training data). The question of interest is to bound the maximum extent to which the risk estimated from this sample under-estimates the true risk.

In general, to obtain bounds on the risk, it is necessary to make some regularity assumptions on the loss. We here require the loss to be Lipshitz in its first argument, and c-bounded. That is, given some prediction function $f : \mathscr{X} \to \mathbb{R}$, we require $|\ell(f(x),y) - \ell(f(x'),y)| \leq \rho|f(x) - f(x')|$ for all x,x', and $|\ell(f(x),y)| \leq c$, for all x. We note that we can obtain the following general bound on c [131]:

$$\begin{aligned}
|\ell(f(x),y)| &= |\ell(f(x),y) - \ell(0,y) + \ell(0,y)| \\
&\leq |\ell(0,y)| + |\ell(f(x),y) - \ell(0,y)| \\
&\leq |\ell(0,y)| + \rho|f(x)| \leq |\ell(0,y)| + \rho LB,
\end{aligned}$$

where $B = \sup_{\theta \in \mathscr{S}} \|\theta\|$, and $L = \sup_{x \in \mathscr{X}} \|\phi(x)\|$. And so, provided $\ell(0,y)$ is finite for all y, boundedness is guaranteed by the Lipschitz property.

By bounding the Rademacher complexity of the class of linear functions induced by encoding, we can apply the results of Section 4.1.2 to obtain bounds on the risk of a model learned from HD representations. The following Lemma gives a bound on this Rademacher complexity:

Lemma 30. *Let $\phi : \mathscr{X} \to \mathscr{H}$ be a VSA encoding, and let $\mathscr{F}_\phi = \{f : f(x) = \langle \theta, \phi(x) \rangle, x \in \mathscr{X}, \theta \in \mathscr{S}\}$. Then:*

$$\mathfrak{R}(\mathscr{F}_\phi \circ \mathscr{D}) \leq \frac{LB}{\sqrt{n}},$$

where $L = \sup_{x \in \mathscr{X}} \|\phi(x_i)\|_2$, and $B = \sup_{\theta \in \mathscr{S}} \|\theta\|$.

The proof is well known in the learning theory literature (e.g. [134]) and simply amounts to bounding the Rademacher complexity of bounded linear functions:

Proof. Let us fix a particular set of samples $\{x_1, ..., x_n\}$. Then, by definition:

$$\begin{aligned}
\mathfrak{R}(\mathscr{F}_\phi \circ \mathscr{D}) &= \mathbb{E}_\sigma \left[\sup_{f \in \mathscr{F}_\phi} \frac{1}{n} \sum_{i=1}^{n} \sigma_i f(x_i) \right] \\
&= \mathbb{E}_\sigma \left[\sup_{\theta \in \mathscr{S}} \left\langle \theta, \frac{1}{n} \sum_{i=1}^{n} \sigma_i \phi(x_i) \right\rangle \right] \\
&\leq \frac{B}{n} \mathbb{E}_\sigma \left[\left\| \sum_{i=1}^{n} \sigma_i \phi(x_i) \right\| \right] \\
&\leq \frac{B}{n} \sqrt{\sum_{i=1}^{n} \|\phi(x_i)\|_2^2} \leq \frac{BL}{\sqrt{n}},
\end{aligned}$$

where $L = \max_{i \in [n]} \|\phi(x_i)\|_2$ and $B = \sup_{\theta \in \mathscr{S}} \|\theta\|_2$. The second-to-last inequality follows by Jensen's inequality and the fact that $\mathbb{E}[\sigma_i \sigma_j] = 1$ if $i = j$ and 0 otherwise. $\square$

As an immediate consequence of the previous result and Theorem 27, we obtain the following risk bound for HD models:

Theorem 31. ***HD Risk Bound***: *Let $\{(x_1, y_1), ..., (x_n, y_n)\}$ be a set of points drawn i.i.d. from a distribution P on $\mathscr{X} \times \mathscr{Y}$. Then, for any particular instantiation of the encoding function, with probability at least $1 - \delta$ over samples of size n, for all $\theta \in \mathscr{S}$:*

$$\mathbb{E}_{(x,y) \sim P}[\ell(\langle \theta, \phi(x) \rangle, y)] \leq \frac{1}{n} \sum_{i=1}^{n} \ell(\langle \theta, \phi(x_i) \rangle, y_i) + \frac{2\rho LB}{\sqrt{n}} + 4c\sqrt{\frac{2 \ln 4/\delta}{n}},$$

where $L = \sup_{x \in \mathscr{X}} \|\phi(x)\|_2$, and $B = \sup_{\theta \in \mathscr{S}} \|\theta\|_2$.

The first term captures the risk on a specific sample, and the second two terms bound the gap between the estimated risk with respect to a sample, and the true risk with respect to the unknown underlying distribution, typically called the "excess risk." The theorem tells us that, to guarantee the excess risk of all models in the hypothesis class induced by encoding is at most ε, it is sufficient to take $n = O(\varepsilon^{-2}\max\{(\rho LB)^2, c^2\ln(4/\delta)\})$. That is to say, this is the maximum number of samples one would need to see in order to guarantee that the true risk of the learned model is within ε of the true risk on the underlying distribution. We note, in particular, that the bound does not directly depend on the dimension at all, only on the norms of $\phi(x)$ and θ. As noted previously, in many cases one can assume $L \approx 1$. However, the encoding dimension is important in determining the first term in the equation, as we now describe.

4.4 The Effect of Encoding Dimension and Precision on Learning with HDC

The result above applies to a particular instantiation of the encoding function. However, in practice, encoding is typically random and approximates some underlying kernel in expectation. Moreover, in practice, it is common to round/threshold the parameter vector θ so as to represent it using lower precision. A natural question is then to bound the maximum possible error between the best possible model obtained from a particular instantiation of the encoding function, and that which could be obtained in expectation over randomness in encoding. Stated another way: we are interested in bounding the discrepancy between models constrained to use d-dimensions and b-bits of precision, and their infinite-dimensional analogues with unconstrained precision. Here we may avail ourselves of some of the work on this topic from the random Fourier features community (e.g. [120]).

In this section, we focus on encoding functions satisfying the conditions of Theorem 29, since this allows us to give a more explicit characterization of the role of encoding dimension. Let us denote by $\mathscr{G}_A$ the set of all functions generated in expectation over randomness in the encoding function. Such functions take the form $g(x) = \sum_{i=1}^{n} \alpha_i \mathbb{E}[\langle \phi(x), \phi(x_i) \rangle] = \sum_{i=1}^{n} \alpha_i k(x, x_i)$. We are interested in bounding the discrepancy between what can be achieved using $\mathscr{G}_A$ as the hypothesis space, and using low dimensional/precision approximations one is limited to in practice.

The following theorem bounds the discrepancy between the best achievable risk using functions from $\mathscr{G}_A$, and that which can be achieved with respect to a specific set of observed data, using d dimensions and restricting θ to use just b-bits of precision per-coordinate.

Theorem 32. *Let $\mathscr{D} = \{(x_1,y_1),...,(x_n,y_n)\}$ be a set of points sampled i.i.d. from a distribution P on $\mathscr{X} \times \mathscr{Y}$, and let $\ell : \mathbb{R}^2 \to \mathbb{R}$ be a ρ-Lipschitz loss function satisfying $|\ell(t,y)| \leq c$, for all $t \in \mathbb{R}$ and $y \in \mathscr{Y}$. Let:*

$$\hat{\theta} = \underset{\theta \in \mathscr{S}_A}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \ell(\langle \theta, \phi(x_i)\rangle, y_i), \text{ and } g^* = \underset{g \in \mathscr{G}_A}{\operatorname{argmin}} \mathbb{E}_{(x,y)\sim P}[\ell(g(x),y)]$$

Then, up to constants, with probability at least $1-\delta$ over randomness in $\mathscr{D}$ and ϕ:

$$\mathbb{E}_{(x,y)}[\ell(\langle \Pi_b(\hat{\theta}), \phi(x)\rangle, y) - \ell(g^*(x),y)] \leq \rho MA \left(\sqrt{\frac{1}{d}\ln\frac{n}{\delta}}\left(1+\frac{1}{2^b-1}\right)\right)\dots$$
$$+\frac{1}{\sqrt{n}}\left(\rho A + c\sqrt{\ln\frac{1}{\delta}}\right),$$

where $\Pi_b(\hat{\theta})$ is the projection of $\hat{\theta}$ onto the set of $\theta \in \mathscr{H}$ representable using b-bits of precision per coordinate.

The proof is rather lengthy and the remainder of the paper will be devoted to developing the intermediate results needed, which are of interest in their own right. The result says that, to guarantee the gap between the risk of a d-dimensional model represented using b-bits of precision, learned from n samples is within ε of the best achievable risk in expectation over randomness in the encoding and using arbitrary precision, it is sufficient to take:

$$d = O\left(\frac{\rho^2 A^2 M^2}{\varepsilon^2}\ln\frac{n}{\delta}\right), n = O\left(\frac{1}{\varepsilon^2}\max\left\{\rho^2 A^2, c^2 \ln\frac{1}{\delta}\right\}\right),$$

and that the loss due to constraining θ to be of low-precision is a lower-order term.

The technique used to prove this result is due to [120], and works roughly as follows. In Theorem 33 we show that, provided that d is chosen suitably large, we can always find a $\theta \in \mathscr{S}_A$

that well approximates the "best" $g \in \mathscr{G}_A$ with respect to a particular sample. In Theorem 34, we extend this result to also hold under constraints on precision. We can then apply the arguments from statistical learning theory introduced above to extend these results, which hold with respect to a specific sample, to hold with respect to the entire data distribution.

4.4.1 Finite Sample Solutions to HD Learning Problems

The main ingredient used to prove Theorem 32 is the following result, which says that, provided d is chosen suitably large, there is always a $\theta \in \mathscr{S}_A$ that well approximates the $g \in \mathscr{G}_A$ achieving minimum loss over a particular sample. In particular:

Theorem 33. *Let $\phi : \mathscr{X} \to \mathscr{H}$ be an encoding approximating a kernel k in the sense of Definition 8. Let $\{(x_1, y_1), ..., (x_n, y_n)\}$ be an arbitrary (not necessarily i.i.d.) set of points, and define:*

$$\hat{g} = \operatorname*{argmin}_{g \in \mathscr{G}_A} \frac{1}{n} \sum_{i=1}^{n} \ell(g(x_i), y_i),$$

for a ρ-Lipschitz loss ℓ. Then there exists $\theta \in \mathscr{S}_A$ such that

$$\frac{1}{n} \sum_{i=1}^{n} \ell(\langle \theta, \phi(x_i) \rangle, y_i) - \ell(\hat{g}(x_i), y_i) \leq \rho A \Delta(d, n).$$

Proof. By the Representer Theorem [80, 129], the loss-minimizing $g \in G_A$ (over a fixed set of n points) can be written in the form:

$$\hat{g}(x_i) = \sum_{j=1}^{n} \alpha_j k(x_i, x_j),$$

for some set of weights $\alpha_1, ..., \alpha_n$. Now, let $\theta = \sum_{i=1}^{n} \alpha_i \phi(x_i)$, whereupon, using the Lipschitz assumption on the loss we conclude:

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n}\ell(\langle\theta,\phi(x_i)\rangle,y_i)-\ell(g(x_i),y_i) &\leq \frac{1}{n}\sum_{i=1}^{n}|\ell(\langle\theta,\phi(x_i)\rangle,y_i)-\ell(g(x_i),y_i)| \\ &\leq \rho\max_{i\in[n]}|\langle\theta,\phi(x_i)\rangle-g(x_i)| \\ &\leq \rho\max_{i\in[n]}\left|\sum_{j=1}^{n}\alpha_j\left(k(x_i,x_j)-\langle\phi(x_i),\phi(x_j)\rangle\right)\right| \\ &\leq \rho A\Delta(d,n),\end{aligned}$$

where $A=\sum_{i=1}^{n}|\alpha_i|$. Therefore:

$$\frac{1}{n}\sum_{i=1}^{n}\ell(\langle\theta,\phi(x_i)\rangle,y_i)-\ell(g(x_i),y_i)\leq\rho A\Delta(d,n).$$

□

The result says that, to ensure there exists a θ in $\mathscr{S}_A$ that achieves to within additive error ε of the best achievable loss using functions in $\mathscr{G}_A$, it is sufficient to take $\Delta(d,n)\leq\varepsilon/\rho A$. In the context of encoding functions satisfying Theorem 29, it is sufficient to take $d=O\left(\frac{\rho^2M^2A^2}{\varepsilon^2}\ln\frac{n}{\delta}\right)$ to guarantee this condition holds with probability at least $1-\delta$ over randomness in the encoding.

HDC and Kernel Methods

In the previous result, functions in $\mathscr{G}_A$ take the form $g(x)=\sum_{i=1}^{n}\alpha_i\mathbb{E}[\langle\phi(x),\phi(x_i)\rangle]=\sum_{i=1}^{n}\alpha_i k(x,x_i)$. Such hypothesis spaces are the basis of kernel methods, of which (kernel) SVMs [148], ridge-regression, and PCA [130] are notable examples. These techniques can all be cast as search problems over hypothesis spaces in the form of $\mathscr{G}_A$, and algorithmic realizations of these techniques typically amount to procedures for fitting the α_i to data [135]. The basic intuition of Theorem 33 is that the loss-minimizing θ, over a finite set of data, achieves within an additive factor, vanishing in d of the best achievable loss using the kernel approximated by encoding.

Kernel methods rely on a very similar intuition to the one underlying HDC: one embeds the data into some high-dimensional space, wherein the dot (inner) product is the salient notion of similarity, meaning linear methods will be effective, when they may not have been originally. HDC differs from conventional realizations of kernel methods in that, in the former, one fits θ directly by materializing $\phi(x)$, whereas in the latter these objects are implicitly defined by k and the α_i. Because kernel methods define the encoding implicitly via k, they can support extremely high-dimension (even infinite dimensional) encodings. However, this comes at the expense of needing to store data to define g, which is problematic in resource constrained settings. HDC does not suffer from this problem since one typically updates θ incrementally as new data is received, which only requires $O(d)$ storage, but this limits one to finite-dimensional models. The gist of Theorem 32 is to bound what is lost by this restriction. This principle of replacing the possibly infinite dimensional feature map of a kernel with a low-dimensional approximation, is the basis of several techniques for kernel approximation in the machine learning literature, notably random Fourier features [119, 120], and the Nyström method [153], which also construct random encodings that approximate kernels and can be used in similar ways.

4.4.2 The Effect of Precision

In practice, one often hopes that θ, ϕ can be represented using low-precision. There are a variety of techniques that are suitable for producing low-precision embeddings [113, 156], but their weighted sum (θ) does not necessarily inherit this characteristic. A natural question is then: how do constraints on the precision of θ effect the achievable risk?

A common approach in the applied literature is to simply quantize using the element-wise sign function. That is, to set $\theta = \text{sign}(\sum_i \alpha_i \phi(x_i))$. However, this approach does not allow one to tune the degree of precision, and does not, in general, approximate the optimal θ. To study the effect of precision in greater generality, we here use a simple randomized rounding rule, which is inspired by [156] who use it to construct low-precision random Fourier features. A similar idea is used in [40] for gradient quantization. This procedure can be used to generate θ's with tunable precision, and which, as we shall see, provably well approximate the full-precision ideal.

An appealing feature of the distributed nature of HD representations is that quantization error can be averaged out over the coordinates in the representation, and so a substantial amount of quantization is possible without adversely affecting the risk.

The technique is simple. Let $x \in [p,q]$ be a real number lying in some interval of width $w = q - p$. We first partition $[p,q]$ into $2^b - 1$ bins of equal size $\varepsilon = w/(2^b - 1)$. Let $p(x), q(x)$ denote the lower and upper endpoints of the bin in which x falls. Then we define the rounding as follows:

$$Q(x) = \begin{cases} p(x) & \text{w.p. } \frac{q(x)-x}{q(x)-p(x)} \\ q(x) & \text{w.p. } \frac{x-p(x)}{q(x)-p(x)}. \end{cases}$$

A short calculation will show that $\mathbb{E}[Q(x)] = x$, and that $|Q(x) - x| \leq w/(2^b - 1)$ (e.g. the quantization error is unbiased and bounded). The following result bounds the maximum error induced by quantization:

Lemma 34. *Let $\mathscr{D} = \{x_1, ..., x_n\}$ be an arbitrary set of points, and let ϕ be an encoding function satisfying the conditions of Theorem 29. For any $\theta \in \{\sum_{i=1}^{n} \alpha_i \phi(x_i) : \|\alpha\|_1 \leq A\}$, with probability at least $1 - \delta$ over randomness in Q, for all $i \in [n]$:*

$$|\langle Q(\theta) - \theta, \phi(x_i)\rangle| \leq \frac{MA}{2^b - 1}\sqrt{\frac{2}{d}\ln\frac{2n}{\delta}},$$

where Q is applied element-wise over θ.

Proof. Let us fix some set of points $\{x_1, ..., x_n\}$ and weights $\alpha_1, ..., \alpha_n$. Then, since Q is unbiased, we have that, for all $i \in [n]$:

$$\mathbb{E}[\langle Q(\theta) - \theta, \phi(x_i)\rangle] = 0,$$

where the expectation is taken with respect to randomness in Q.

Fix some $j \in [d]$. By the assumption that $|\phi(x)_i| \leq \sqrt{M/d}$, we conclude $\left|\sum_i \alpha_i \phi(x_i)_j\right| \leq$

$A\sqrt{M/d}$, where $A = \sum_i |\alpha_i|$. Therefore, the maximum quantization error per-coordinate is:

$$\left|Q(\theta)_j - \theta_j\right| \leq \frac{A\sqrt{M}}{\sqrt{d}(2^b-1)}.$$

Then, by Hoeffding's inequality and the union bound over all n points, for any $t > 0$:

$$\begin{aligned}\Pr\left(\exists i \text{ s.t. } |\langle Q(\theta)-\theta, \phi(x_i)\rangle| \geq t\right) &\leq 2n\exp\left(-\frac{2(2^b-1)^2 t^2 d}{(2MA)^2}\right)\\ &= 2n\exp\left(-\frac{(2^b-1)^2 t^2 d}{2M^2A^2}\right)\end{aligned}$$

Solving the bound for an arbitrary error-threshold δ yields the result. □

An immediate corollary of the result is that, for any ρ-Lipschitz loss function, and any set of points $\mathscr{D} \subset (\mathscr{X} \times \mathscr{Y})^n$, there exists $\hat{\theta}$ such that:

$$\left|\frac{1}{n}\sum_{i=1}^{n} \ell(\langle \Pi_b(\hat{\theta}), \phi(x_i)\rangle, y_i) - \ell(\hat{g}(x_i), y_i)\right| \leq \rho MA\sqrt{\frac{1}{d}\ln\frac{2n}{\delta}}\left(1+\frac{1}{2^b-1}\right),$$

where $\hat{g}$ minimizes the loss over $\mathscr{D}$, and we have defined $\Pi_b(\theta) = Q(\theta)$. That is, there exists a low-precision parameter vector that achieves within an additive factor (that can be made as small as desired by choosing d sufficiently large), of what can be achieved in expectation over randomness in the encoding. We emphasize that this result does not require an i.i.d. assumption on the data.

4.5 Proof of Theorem 32

Armed with the preceding results, we may prove Theorem 32. The proof requires the following technical Lemmas.

Lemma 35. *Let $\mathscr{D} = \{x_1, ..., x_n\}$ be any fixed set of n points, and let $\phi : \mathscr{X} \to \mathscr{H}$ be an encoding approximating a kernel k in the sense of Theorem 29. Then, for any $\theta \in \{\sum_{i=1}^n \alpha_i \phi(x_i) : \|\alpha\|_1 \leq$*

$A\}$, there exists $g \in \mathscr{G}_A$ such that, for all $i \in [n]$, with probability at least $1-\delta$ over randomness in the encoding:

$$|\langle \theta, \phi(x_i)\rangle - g(x_i)| \leq 2MA\sqrt{\frac{1}{d}\ln\frac{n}{\delta}}.$$

Proof. By definition:

$$\theta = \sum_{j=1}^{n} \alpha_j \phi(x_j),$$

for some set of weights $\alpha_1, ..., \alpha_n$. Now, let us take:

$$g(x_i) = \sum_{j=1}^{n} \alpha_j k(x_i, x_j).$$

Whereupon, by Theorem 29, for all $i \in [n]$:

$$\begin{aligned}
|\langle \phi(x_i), \theta\rangle - g(x_i)| &= \left|\sum_{j=1}^{n} \alpha_j \left(\langle \phi(x_i), \phi(x_j)\rangle - k(x_i, x_j)\right)\right| \\
&\leq \sum_{j=1}^{n} \left|\alpha_j \left(\langle \phi(x_i), \phi(x_j)\rangle - k(x_i, x_j)\right)\right| \\
&\leq A \max_{i\in[n]} \left|\left(\langle \phi(x_i), \phi(x_j)\rangle - k(x_i, x_j)\right)\right| \\
&\leq 2MA\sqrt{\frac{1}{d}\ln\frac{n}{\delta}},
\end{aligned}$$

as claimed. □

The next Lemma bounds the Rademacher complexity of the hypothesis space induced by HD encoding in terms of the Rademacher complexity of the RKHS for the kernel approximated by encoding:

Lemma 36. *Let $\phi : \mathscr{X} \to \mathscr{H}$ be a VSA encoding satisfying the conditions of Theorem 29 for a kernel k on $\mathscr{X}$, and let $\{x_1, ..., x_n\}$ be any set of n points. Define :*

$$\mathscr{S}_A^n = \left\{\sum_{i=1}^{n} \alpha_i \phi(x_i) : \sum_{i=1}^{n} |\alpha_i| \leq A\right\},$$

let $\mathscr{F}_\phi = \{f : f(x) = \langle \theta, \phi(x)\rangle, \theta \in \mathscr{S}_A^n\}$*, and let* $\mathscr{F}_{\phi,b} = \{f : f(x) = \langle \Pi_b(\theta), \phi(x)\rangle, \theta \in \mathscr{S}_A^n\}$*, where* Π_b *is as defined in Theorem 34. Then, for any sample* $\{x_1, ..., x_n\}$*, with probability at least* $1-\delta$ *over randomness in the encoding:*

$$\mathfrak{R}(\mathscr{F}_{\phi,b} \circ \mathscr{D}) \leq \mathfrak{R}(\mathscr{F}_\phi \circ \mathscr{D}) + \frac{2MA}{2^b-1}\sqrt{\frac{1}{d}\ln\frac{2n}{\delta}} \leq \frac{A}{\sqrt{n}} + 2MA\sqrt{\frac{1}{d}\ln\frac{2n}{\delta}}\left(1+\frac{1}{2^b-1}\right).$$

Proof. Let us fix some set of n points $\{x_1, ..., x_n\}$. By definition:

$$\begin{aligned}
\mathfrak{R}(\mathscr{F}_{\phi,b} \circ \mathscr{D}) &= \mathbb{E}_\sigma\left[\sup_{f_b \in \mathscr{F}_{\phi,b}} \frac{1}{n}\sum_{i=1}^n \sigma_i f_b(x_i)\right] \\
&= \mathbb{E}_\sigma\left[\sup_{\theta \in \mathscr{S}_A} \frac{1}{n}\sum_{i=1}^n \sigma_i\left(\langle \Pi_b(\theta), \phi(x_i)\rangle + \langle \theta, \phi(x_i)\rangle - \langle \theta, \phi(x_i)\rangle\right)\right] \\
&= \mathbb{E}_\sigma\left[\sup_{\theta \in \mathscr{S}_A} \frac{1}{n}\sum_{i=1}^n \sigma_i \langle \theta, \phi(x_i)\rangle\right] + \mathbb{E}_\sigma\left[\sup_{\theta \in \mathscr{S}_A} \frac{1}{n}\sum_{i=1}^n \sigma_i \langle \phi(x_i), \Pi_b(\theta) - \theta\rangle\right] \\
&\leq \mathbb{E}_\sigma\left[\sup_{f \in \mathscr{F}_\phi} \frac{1}{n}\sum_{i=1}^n \sigma_i f(x_i)\right] + \frac{MA}{2^b-1}\sqrt{\frac{2}{d}\ln\frac{2n}{\delta}}. \\
&= \mathfrak{R}(\mathscr{F}_\phi \circ \mathscr{D}) + \frac{MA}{2^b-1}\sqrt{\frac{2}{d}\ln\frac{2n}{\delta}},
\end{aligned}$$

where we have used Lemma 34 in going from the second to last inequality to the last. By a similar line of reasoning:

$$\begin{aligned}
\mathfrak{R}(\mathscr{F}_\phi \circ \mathscr{D}) &= \mathbb{E}_\sigma\left[\sup_{f \in \mathscr{F}_\phi} \frac{1}{n}\sum_{i=1}^n \sigma_i f(x_i)\right] \\
&= \mathbb{E}_\sigma\left[\sup_{\alpha : \|\alpha\|_1 \leq A} \frac{1}{n}\sum_{i=1}^n \sigma_i\left(\sum_{j=1}^n \alpha_j(\langle \phi(x_i), \phi(x_j)\rangle)\right)\right] \\
&= \mathbb{E}_\sigma\left[\sup_{\alpha : \|\alpha\|_1 \leq A} \frac{1}{n}\sum_{i=1}^n \sigma_i\left(\sum_{j=1}^n \alpha_j \langle \phi(x_i), \phi(x_j)\rangle + k(x_i, x_j) - k(x_i, x_j)\right)\right] \\
&= \mathfrak{R}(\mathscr{G}_A \circ \mathscr{D}) + \mathbb{E}_\sigma\left[\sup_{\alpha : \|\alpha\|_1 \leq A} \frac{1}{n}\sum_{i=1}^n \sigma_i\left(\sum_{j=1}^n \alpha_j \langle \phi(x_i), \phi(x_j)\rangle - k(x_i, x_j))\right)\right] \\
&\leq \mathfrak{R}(\mathscr{G}_A \circ \mathscr{D}) + 2MA\sqrt{\frac{1}{d}\ln\frac{n}{\delta}},
\end{aligned}$$

where we have used Lemma 35 for the last inequality. The last step is to bound $\mathfrak{R}(\mathscr{G}_A \circ \mathscr{D})$. By a standard result concerning the Rademacher complexity of RKHS functions (e.g. [135]):

$$\mathfrak{R}(\mathscr{G}_A \circ \mathscr{D}) \leq \frac{B\sqrt{\mathrm{Tr}(K)}}{n},$$

where $K_{ij} = k(x_i, x_j)$, and:

$$B = \sup_{\alpha : \|\alpha\|_1 \leq A} \sqrt{\sum_{ij} \alpha_i \alpha_j k(x_i, x_j)} \leq \sqrt{\sum_{ij} |\alpha_i||\alpha_j|} \leq A,$$

Since we assume $|k(x,x')| \leq 1$. Moreover, $\mathrm{Tr}(K) = n$, whereupon

$$\begin{aligned}
\mathfrak{R}(\mathscr{F}_{\phi,b} \circ \mathscr{D}) &\leq \mathfrak{R}(\mathscr{F}_{\phi} \circ \mathscr{D}) + \frac{MA}{2^b - 1}\sqrt{\frac{2}{d}\ln\frac{2n}{\delta}} \\
&\leq \mathfrak{R}(\mathscr{G}_A \circ \mathscr{D}) + 2MA\sqrt{\frac{1}{d}\ln\frac{2n}{\delta}}\left(1 + \frac{1}{2^b - 1}\right) \\
&\leq \frac{A}{\sqrt{n}} + 2MA\sqrt{\frac{1}{d}\ln\frac{2n}{\delta}}\left(1 + \frac{1}{2^b - 1}\right)
\end{aligned}$$

as claimed. □

At long last, we prove Theorem 32:

Proof of Theorem 32. First expand:

$$L_P(\Pi_b(\hat{\theta}_n)) - L_P(g^*) = (L_P(\Pi_b(\hat{\theta}_n)) - L_n(\hat{\theta}_n)) + (L_n(\hat{\theta}_n) - L_P(g^*)). \tag{4.4}$$

To bound the first term, we do another decomposition:

$$L_P(\Pi_b(\hat{\theta}_n)) - L_n(\hat{\theta}_n) = \underbrace{\left(L_P(\Pi_b(\hat{\theta}_n)) - L_n(\Pi_b(\hat{\theta}_n))\right)}_{(1)} + \underbrace{\left(L_n(\Pi_b(\hat{\theta}_n)) - L_n(\hat{\theta}_n)\right)}_{(2)}.$$

Term (1) is bounded via Theorem 27 and Lemma 36 as:

$$L_P(\Pi_b(\hat{\theta}_n)) - L_n(\Pi_b(\hat{\theta}_n)) \leq \frac{2\rho A}{\sqrt{n}} + 4\rho MA\sqrt{\frac{1}{d}\ln\frac{2n}{\delta}}\left(1+\frac{1}{2^b-1}\right) + 4c\sqrt{\frac{2\ln(4/\delta)}{n}}$$

Term (2) is bounded via Theorem 34 as:

$$L_n(\Pi_b(\hat{\theta}_n)) - L_n(\hat{\theta}_n) \leq \rho MA\sqrt{\frac{2}{d}\ln\frac{2n}{\delta}}\left(1+\frac{1}{2^b-1}\right)$$

Therefore, the first term in Equation 4.4 is bounded by:

$$L_P(\Pi_b(\hat{\theta}_n)) - L_n(\hat{\theta}_n) \leq \frac{2\rho A}{\sqrt{n}} + 6\rho MA\sqrt{\frac{1}{d}\ln\frac{2n}{\delta}}\left(1+\frac{1}{2^b-1}\right) + 4c\sqrt{\frac{2\ln(4/\delta)}{n}}$$

To bound the second term in Equation 4.4 we do another decomposition:

$$L_P(\hat{\theta}_n) - L_P(g^*) = \underbrace{\left(L_n(\hat{\theta}_n) - L_n(\hat{g}_n)\right)}_{(1)} + \underbrace{\left(L_n(\hat{g}_n) - L_P(g^*)\right)}_{(2)},$$

and proceed term by term. Term (1) is bounded via Theorem 33 as:

$$L_n(\hat{\theta}_n) - L_n(\hat{g}_n) \leq 2\rho MA\sqrt{\frac{1}{d}\ln\frac{n}{\delta}}$$

Term (2) is bounded via Theorem 28 and Lemma 36 as:

$$L_n(\hat{g}_n) - L_P(g^*) \leq \frac{2\rho A}{\sqrt{n}} + 5c\sqrt{\frac{2\ln(8/\delta)}{n}}.$$

Putting these together we obtain that the second term in Equation 4.4 is bounded by

$$L_P(\hat{\theta}_n) - L_P(g^*) \leq \frac{2\rho A}{\sqrt{n}} + 2\rho MA\sqrt{\frac{1}{d}\ln\frac{n}{\delta}} + 5c\sqrt{\frac{2\ln(8/\delta)}{n}}.$$

Finally, combining both bounds, we obtain that:

$$L_P(\Pi_b(\hat{\theta}_n)) - L_P(g^*) \leq 8\rho MA\left(\sqrt{\frac{1}{d}\ln\frac{2n}{\delta}}\left(1+\frac{1}{2^b-1}\right)\right) + \frac{4}{\sqrt{n}}\left(\rho A + 9c\sqrt{\ln\frac{8}{\delta}}\right).$$

completing the proof. □

4.6 Conclusion

This chapter develops a formal perspective on learning with HDC. We extend the classic capacity theory of HDC, which has focused on the problem of encoding and decoding specific data items, to the learning setting, which focuses on the problem of prediction, using techniques from the literature on statistical learning theory and kernel methods. In particular, we use the notion of statistical risk and provide bounds on the dimension and precision required of the representations to achieve within an additive factor of the best achievable model in expectation over randomness in the encoding.

On a more practical note, we cast a general approach to learning with VSAs, in which one fits linear models to encoded versions of data, as an approximate form of kernel machine, similar to techniques like random Fourier features and the Nyström method which are widely used in the machine learning community. While our perspective is applicable to a reasonably broad family of VSAs and practically relevant approaches to learning, there remain many open questions. In particular, there are other relevant paradigms for learning, like using VSA encodings as inputs to neural networks [105, 4], which are not interpretable as kernel machines, and the statistical model used to assess learning capacity here is not always appropriate. In particular, it would be of interest to study these models in the online setting which does not impose distributional assumptions.

Chapter 4 contains material from "A Formal Perspective on Learning with Vector Symbolic Architectures," by Anthony Thomas, Sanjoy Dasgupta, Tara Javidi, and Tajana Rosing, as

it was submitted to the 2023 Conference on Neural Information Processing Systems (NeurIPS). The dissertation author was the primary investigator and author of this paper.

Chapter 5

Summary and Future Directions

Biological brains remain, by a wide margin, the most capable platforms for effecting cognition. Even small insects like honey bees and fruit flies are capable of performing sophisticated learning and reasoning tasks that exceed the capabilities of modern artificial intelligence [7, 27], and do so using “hardware” that requires vanishingly little energy, is intrinsically fault-tolerant, and capable of self-repair (and self-replication). Motivated by these remarkable capabilities, all of which are desirable for conventional computing systems, there has been substantial interest in developing computing paradigms motivated by biology [122, 125, 59, 70, 39]. Hyperdimensional computing is an effort in this direction that focuses primarily on computing using distributed representations of data. The basic mechanisms used in HDC are simple to understand, remarkably versatile, and the technique has demonstrated empirical success as a mechanism to effect various cognitive information processing tasks in resource constrained settings [116, 115, 85, 93].

5.1 Thesis Summary

The focus of this dissertation has been on developing the theoretical foundations of HDC, and in particular, on developing a rigorous and general understanding of the conditions under which different kinds of information processing tasks posed on HD representations of data will succeed. A key contribution of this work has been the development of analysis that holds in the finite dimensional setting, in the sense that it does not involve asymptotic approximations or statements that hold only in expectation, and is general, in the sense that it is reasonably agnostic to the details of a particular architecture.

The most basic question in HDC is to understand how data items and collections thereof can be represented in a manner that permits reliable recovery. In the first part of Chapter 2, we develop a novel mathematical framework, based on the notion of incoherence from the compressed sensing literature [43], that allows us to give general sufficient conditions under which a variety of different procedures for encoding and decoding discrete data will succeed, even in the presence of noise. While some of these questions have been studied in prior work under specific assumptions about the encoding mechanism [105, 53] or using asymptotic approximations [48], our formalism allows us to decouple the analysis of decoding and noise robustness from any particular method for generating the encodings, and readily yields formal guarantees in the finite-dimensional setting one is limited to in practice. The remainder of the chapter is devoted to analyzing different techniques for representing data in a Euclidean space. In this context, rather than asking the codewords to be incoherent (i.e. almost-orthogonal), one wants the degree of coherence to reflect some underlying notion of similarity in the data space. We calculate this similarity function for several different encoding functions of interest, and provide bounds on the fidelity with which it is approximated by dot-products in HD space. The chapter concludes by discussing some implications for learning from HD representations, and in particular, give sufficient conditions for the preservation of cluster structure and linear separability.

The encoding schemes discussed in the previous chapter are versatile and remarkably effective for a range of different tasks. However, basic instantiations of these techniques encounter serious bottlenecks when the input data itself is high-dimensional. For instance, to represent data drawn from an alphabet of m symbols, the conventional approach is to store a d-dimensional codeword corresponding to each symbol, which is prohibitive when m and/or d are large. In chapter 3, we analyze approaches based on hashing, which allow one to generate encodings "on-the-fly" without needing to store a codebook. While there has been a substantial body of work on developing techniques for codeword re-materialization [77, 81, 116], the bulk of this work has focused on settings in which one wishes to repeatedly generate the same sequence of pseudo-random codewords repeatedly in the same order (e.g. to encode sequential data using the "position-ID" method from chapter 3). Hashing methods, by contrast, are performant in the setting that the data to encode arrives in a non-deterministic order, and comfortably scale to very high-dimensional inputs. Building on ideas from the previous chapter, we provide novel analyses of sparse and hashing based encoding methods for use in learning applications, and show formally that these techniques enjoy similar guarantees while being substantially more efficient. These techniques are also appealing because they naturally lead to sparse and binary encodings, without the need for ad-hoc quantization encountered in dense representations. We validate this theory experimentally on a popular large-scale classification benchmark [34], and with an implementation on an FPGA, demonstrating that the hashing methods studied in this chapter offer an over $100\times$ speedup compared to other comparable encoding techniques [45].

Recent years have seen substantial interest in using HD representations of data in learning algorithms (e.g. [62, 116, 85] to name but a few). The basic argument of this line of work is that the distributed nature of HD representations aligns naturally with highly parallel hardware platforms like FPGAs [127, 77] and in-memory architectures [76], and can be used to effect extremely efficient alternatives to "conventional" approaches to learning like SVMs [93] and certain types of trainable neural network [62]. However, most theoretical work on HDC has focused on analyzing the ability different HD architectures to store and recall specific data items, often called "storage capacity." While Chapters 2 and 3 introduced some preliminary results concerning the use of HD representations in learning algorithms, they left open several important questions. In particular, the results presented in those chapters only addressed the problem of learning linear separators, and assume the data is separable in its ambient representation, which is restrictive in practice. In Chapter 4 we develop a significantly more general view of learning from HD representations.

Building on ideas from Chapter 2, we interpret the encoding function as approximating a particular similarity function (i.e. a kernel) on $\mathscr{X}$ via a dot-product in $\mathscr{H}$. This is significant, because most approaches to learning used in practice amount to fitting linear functions in HD space, and the space of all such functions is completely determined by the kernel induced by encoding. Intuitively, the encoding function can be viewed as a noisy approximation to some underlying kernel of interest, which encodes prior beliefs about what notion of similarity is salient for the data at hand. The operative question is then to determine what level of noise can be tolerated. To make progress in this direction, we extend the classic "capacity theory" of HDC/VSAs [105, 144, 48] to the learning setting using the notion of risk from statistical learning theory [149, 10], which measures the expected discrepancy–as measured by some loss function–between the predictions generated by an HD model and the ground truth. We provide bounds on the encoding dimension, number of training samples, and precision that are sufficient to guarantee the best achievable risk of models learned from HD encodings is within an additive factor ε of what can be achieved in expectation over randomness in the encoding, and with perfect knowledge of the underlying data distribution. Our work also elucidates the connections between HDC and kernel methods, an influential line of work that is also centered around learning from high-dimensional representations of data. In particular, in much the same way as techniques like random Fourier features [119], fitting linear functions in HD space can be interpreted as approximating kernel machines fit on the ambient representation of the data using the kernel induced by encoding. This observation is useful both for interpreting the predictions generated by HD models, but also because the literature on kernel methods has proposed a wide range of algorithms for learning problems that can be adapted to the HD setting.

5.2 Future Directions

In general, a fundamental goal of HDC is the development of cognitive systems that: (1) run on very low-power devices, (2) have rigorous guarantees on correctness, and (3) continuously adapt to a changing environment. The hardware focused literature has made significant strides in meeting the first goal, and the theoretical literature (including this dissertation) has made progress on the second in some settings. However, the third goal remains largely aspirational. Meeting this objective will require the development of encoding techniques and learning algorithms, that are both plausible for use in HD computing and are able to adapt to a changing environment. In the following sections, I outline two important directions in this regard.

5.2.1 Online and Low-Precision Algorithms for Basic Learning Tasks

The overwhelming majority of work on learning with HDC has focused on supervised classification problems, and uses a traditional approach to learning in which one fits a model using training data, and then assesses its performance on a held-out test sample. The validity of this approach is predicated on the assumption that the test set is representative of the actual environment in which the system will be deployed. However, real environments are usually dynamic, and a static model is unlikely to remain performant in perpetuity.

The statistical learning model studied in Chapter 4 is not well suited to this setting, because it fundamentally assumes a static distribution for the data generating process. A more realistic approach is the online learning model [23]. In the online model, learning is viewed as a series of rounds played between a learning agent, and a (possibly malicious) environment. During each round $t = 1,2,...$ the learner proposes a model $f_t : \mathscr{X} \to \mathscr{Y}$, the environment then presents a test point x_t, y_t, and the learner incurs a loss $\ell(f_t(x_t), y_t)$. At the conclusion of the round, the learner is allowed to update their model in an effort to decrease the loss on subsequent rounds. The goal of the learner is to minimize their regret, which is defined as the gap between the cumulative loss incurred using the online updates, and the best achievable loss in hindsight (e.g. what could have been obtained by storing all data and directly minimizing the loss function):

$$R_T = \sum_{t=1}^{T} \ell(f_t(x_t), y_t) - \min_{f \in \mathscr{F}} \sum_{t=1}^{T} \ell(f(x_t), y_t).$$

There are extensions to this basic notion of regret that allow one to describe situations in which the “best” model in hindsight changes over time [58]. The online framework is appealing because it leads to algorithms that update models incrementally, and does not distinguish between a “training” and “testing” phase. That is, the model is constantly being evaluated against new data, which is not assumed to come from any particular distribution, and may, in fact, be chosen maliciously to try to yield the largest possible loss.

There is a substantial body of work devoted to online learning [23, 58], and the Perceptron algorithm, which is widely used in HDC, is a canonical online algorithm for classification [134]. This literature is most well developed in the context of supervised learning problems, and in particular those which can be described as convex optimization problems. However, this literature generally still assumes access to labeled training data, and assumes unbounded memory/precision, both of which are problematic for deployments “in the wild” on resource constrained devices. Thus, an important direction of future work will be to develop bounded-memory online algorithms for basic learning tasks, focusing in particular on unsupervised tasks like density estimation, principal components analysis, and clustering. While these tasks have been the focus of a vast body of research in general, the literature is thinner in the online setting, and thinner still in the online setting with a memory budget.

5.2.2 Adaptive Encodings

The encoding methods discussed in this work are all non-adaptive. That is, one fixes the encoding a-priori of any particular task, and it is static throughout the lifetime of the system. This is, in my impression, the dominant approach in practice as well. This approach is appealing because it disentangles encoding and learning, which greatly simplifies analysis and implementation. Moreover, as the previous discussion has made clear, even static and random encodings can be quite powerful in some settings. However, the success of this approach relies heavily on strong prior knowledge about what facets of the data are salient for a particular task, and choosing a bad encoding can cripple performance on downstream tasks. This suggests that the encoding process should be at least somewhat adaptive to changes in the underlying data, and robust to mis-specification. Current efforts to introduce adaptivity typically amount to treating the encoding operation as a type of trainable neural network and fitting its parameters using SGD [155, 95], or using some pre-trained deep architecture as a feature extractor [44]. However, this approach would seem to lose much of the simplicity and rigor that makes HDC appealing in the first place, and simply re-casts it as yet another neural network architecture.

One plausible alternative is to develop encoding techniques that are adaptive to geometric structure in data. Many types of real world data are collected in a high-dimensional setting, but are constrained by physical reality in such a way that they can be described by a much lower-dimensional set of parameters. For instance, imagine a video of a ball thrown through 3D space. The raw data is a sequence of video frames, which are themselves high-dimensional arrays. However, the path of the ball is not arbitrary, and can be accurately described by a handful of parameters (e.g. velocity at release, wind-speed, etc...). A common way to model this situation is to assume the data lies on a low-dimensional manifold (a curved surface that is locally flat), that has been embedded in a high-dimensional Euclidean space [46].

There is a very large literature in machine learning and applied mathematics devoted to this problem [43, 11, 31], which has led to a number of useful algorithms for clustering [136, 98], and semi-supervised learning (e.g. learning from limited data) problems [20, 158]. However, this literature cannot be imported directly into the HD setting because algorithms are typically formulated in the batch setting, and around the high-precision ambient representations of data. Nonetheless, there is a close connection between this literature and HDC that again makes use of the kernel interpretation of the encoding function.

Given a set of n points $\mathscr{D} = \{x_1, ..., x_n\}$ lying on a manifold $\mathscr{M} \subset \mathbb{R}^m$, many of these techniques are based on the spectral decomposition a graph described by an adjacency matrix $A \in \mathbb{R}^{n \times n}$, where $A_{ij} = s(x_i, x_j)$ (or more formally the Laplacian associated with this matrix), for s a non-negative similarity function. Letting $\Phi(\mathscr{D}) \in \mathbb{R}^{n \times d}$ denote the matrix whose rows correspond to encodings (e.g. $\phi(x_i)$) of points in $\mathscr{D}$, we can interpret the matrix $\tilde{A} = \Phi(\mathscr{D})\Phi(\mathscr{D})^T$ as the adjacency matrix of a graph on $\mathscr{D}$ whose similarity function is $s(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$

(where we continue to require this similarity be non-negative). From this, we can recover many of the spectral methods for manifold learning described in the paper above directly from the HD representations of points. However, a more serious problem is that the approach described above is still in the batch setting, which necessitates the development of online variants of the algorithms described above. There is very little work on studying manifold learning in the online setting, and developing suitable approaches or analysis for this setting would be a significant contribution even outside of HDC.

Bibliography

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: a system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.

[2] Dimitris Achlioptas. Database-friendly random projections. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 274–281, 2001.

[3] U.S. Energy Information Administration. Electric power monthly: Table 5.6.a. average price of electricity to ultimate customers by end-use sector. https://www.eia.gov/electricity/monthly/epm_table_grapher.php?t=epmt_5_6_a, March, 2023.

[4] Pedro Alonso, Kumar Shridhar, Denis Kleyko, Evgeny Osipov, and Marcus Liwicki. Hyperembed: Tradeoffs between resources and performance in NLP tasks with hyperdimensional computing enabled embedding of n-gram statistics. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2021.

[5] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122, 2008.

[6] Austin Appleby. Murmur3 hash. https://github.com/aappleby/smhasher, 2016.

[7] Aurore Avarguès-Weber, Adrian G Dyer, Maud Combe, and Martin Giurfa. Simultaneous mastering of two abstract concepts by the miniature brain of bees. *Proceedings of the National Academy of Sciences*, 109(19):7481–7486, 2012.

[8] Baktash Babadi and Haim Sompolinsky. Sparseness and expansion in sensory representations. *Neuron*, 83(5):1213–1226, 2014.

[9] Andrew R Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, 1993.

[10] Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

[11] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15:1373–1396, 2003.

[12] A.J. Bell and T.J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

[13] Christopher M. Bishop. *Pattern recognition and machine learning, 5th Edition*. Information science and statistics. Springer, 2007.

[14] Burton H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[15] Stephen P. Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2014.

[16] Andrei Broder and Michael Mitzenmacher. Network applications of bloom filters: A survey. *Internet Mathematics*, 1(4):485–509, 2004.

[17] Andrei Z Broder. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*, pages 21–29. IEEE, 1997.

[18] Alessio Burrello, Kaspar Schindler, Luca Benini, and Abbas Rahimi. One-shot learning for ieeg seizure detection using end-to-end binary operations: Local binary patterns with hyperdimensional computing. In *2018 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 1–4. IEEE, 2018.

[19] Alessio Burrello, Kaspar Anton Schindler, Luca Benini, and Abbas Rahimi. Hyperdimensional computing with local binary patterns: One-shot learning for seizure onset detection and identification of ictogenic brain regions from short-time iEEG recordings. *IEEE Transactions on Biomedical Engineering*, 2019.

[20] Jeff Calder, Brendan Cook, Matthew Thorpe, and Dejan Slepcev. Poisson learning: Graph based semi-supervised learning at very low label rates. In *International Conference on Machine Learning*, pages 1306–1316. PMLR, 2020.

[21] Sophie JC Caron, Vanessa Ruta, LF Abbott, and Richard Axel. Random convergence of

olfactory inputs in the drosophila mushroom body. *Nature*, 497(7447):113–117, 2013.

[22] Nicolò Cesa-Bianchi and Claudio Gentile. Improved risk tail bounds for on-line algorithms. In *Advances in Neural Information Processing Systems*, volume 18, 2005.

[23] Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

[24] Maurice J Chacron, Andre Longtin, and Leonard Maler. Efficient computation via sparse coding in electrosensory neural networks. *Current Opinion in Neurobiology*, 21(5):752–760, 2011.

[25] Olivier Chapelle, Eren Manavoglu, and Romer Rosales. Simple and scalable response prediction for display advertising. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(4):1–34, 2014.

[26] Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *International conference on machine learning*, pages 2285–2294. PMLR, 2015.

[27] Lars Chittka. *The mind of a bee*. Princeton University Press, 2022.

[28] Krzysztof M Choromanski, Mark Rowland, and Adrian Weller. The unreasonable effectiveness of structured random orthogonal embeddings. In *Advances in Neural Information Processing Systems*, pages 219–228, 2017.

[29] Kenneth L Clarkson, Shashanka Ubaru, and Elizabeth Yang. Capacity analysis of vector symbolic architectures. *arXiv preprint arXiv:2301.10352*, 2023.

[30] Michael B Cohen, TS Jayram, and Jelani Nelson. Simple analyses of the sparse Johnson-Lindenstrauss transform. In *1st Symposium on Simplicity in Algorithms (SOSA 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[31] Ronald R Coifman and Stéphane Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30, 2006.

[32] Graham Cormode and Shan Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

[33] Xilinx Corporation. Vitis high-level synthesis user guide (ug1399). https://docs.xilinx.com/r/en-US/ug1399-vitis-hls/Introduction, June 2022.

[34] Criteo Research. Criteo research datasets. https://ailab.criteo.com/ressources/, 2021. Accessed: September, 2021.

[35] G. Cybenko. Approximations by superpositions of sigmoidal functions. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, 1989.

[36] Anirban Dasgupta, Ravi Kumar, and Tamás Sarlós. A sparse Johnson-Lindenstrauss transform. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 341–350, 2010.

[37] Sanjoy Dasgupta, Timothy C Sheehan, Charles F Stevens, and Saket Navlakha. A neural data structure for novelty detection. *Proceedings of the National Academy of Sciences*, 115(51):13093–13098, 2018.

[38] Sanjoy Dasgupta and Christopher Tosh. Expressivity of expand-and-sparsify representations. *arXiv preprint arXiv:2006.03741*, 2020.

[39] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99, 2018.

[40] Christopher De Sa, Megan Leszczynski, Jian Zhang, Alana Marzoev, Christopher R Aberger, Kunle Olukotun, and Christopher Ré. High-accuracy low-precision training. *arXiv preprint arXiv:1803.03383*, 2018.

[41] Aditya Desai, Yanzhou Pan, Kuangyuan Sun, Li Chou, and Anshumali Shrivastava. Semantically constrained memory allocation (SCMA) for embedding in efficient recommendation systems. *arXiv preprint arXiv:2103.06124*, 2021.

[42] David L Donoho, Michael Elad, and Vladimir N Temlyakov. Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Transactions on Information Theory*, 52(1):6–18, 2005.

[43] David L Donoho and Carrie Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences*, 100(10):5591–5596, 2003.

[44] Arpan Dutta, Saransh Gupta, Behnam Khaleghi, Rishikanth Chandrasekaran, Weihong Xu, and Tajana Rosing. Hdnn-pim: Efficient in memory design of hyperdimensional computing with feature extraction. In *Proceedings of the Great Lakes Symposium on VLSI 2022*, pages 281–286, 2022.

[45] Manuel Eggimann, Abbas Rahimi, and Luca Benini. A 5 μw standard cell memory-based configurable hyperdimensional computing accelerator for always-on smart sensing. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 68(10):4116–4128, 2021.

[46] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, 2016.

[47] E Paxon Frady, Denis Kleyko, Christopher J Kymn, Bruno A Olshausen, and Friedrich T Sommer. Computing on functions using randomized vector representations. *arXiv preprint arXiv:2109.03429*, 2021.

[48] E Paxon Frady, Denis Kleyko, and Friedrich T Sommer. A theory of sequence indexing and working memory in recurrent neural networks. *Neural Computation*, 30(6):1449–1513, 2018.

[49] Edward Paxon Frady, Denis Kleyko, and Friedrich T Sommer. Variable binding for sparse distributed representations: theory and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[50] David A Freedman. On tail probabilities for martingales. *the Annals of Probability*, pages 100–118, 1975.

[51] Daichi Fujiki, Scott Mahlke, and Reetuparna Das. In-memory data parallel processor. *ACM SIGPLAN Notices*, 53(2):1–14, 2018.

[52] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980.

[53] Stephen I Gallant and T Wendy Okaywe. Representing objects, relations, and sequences. *Neural Computation*, 25(8):2038–2078, 2013.

[54] Ross W Gayler. Vector symbolic architectures answer jackendoff's challenges for cognitive neuroscience. *arXiv preprint cs/0412059*, 2004.

[55] Saransh Gupta, Mohsen Imani, and Tajana Rosing. Felix: Fast and energy-efficient logic in memory. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–7. IEEE, 2018.

[56] David Hanson and Farroll Wright. A bound on tail probabilities for quadratic forms in independent random variables. *The Annals of Mathematical Statistics*, 42(3):1079–1083, 1971.

[57] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[58] Elad Hazan. Introduction to online convex optimization. *Foundations and Trends® in*

Optimization, 2(3-4):157–325, 2016.

[59] John Hertz, Anders Krogh, Richard G Palmer, and Heinz Horner. Introduction to the theory of neural computation. *Physics Today*, 44(12):70, 1991.

[60] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008.

[61] Paul Honeine and Cédric Richard. The angular kernel in machine learning for hyperspectral data classification. In *2010 2nd Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing*, pages 1–4. IEEE, 2010.

[62] Mohsen Imani, Deqian Kong, Abbas Rahimi, and Tajana Rosing. Voicehd: Hyperdimensional computing for efficient speech recognition. In *2017 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–8. IEEE, 2017.

[63] Mohsen Imani, John Messerly, Fan Wu, Wang Pi, and Tajana Rosing. A binary learning framework for hyperdimensional computing. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 126–131. IEEE, 2019.

[64] Mohsen Imani, Justin Morris, Samuel Bosch, Helen Shu, Giovanni De Micheli, and Tajana Rosing. Adapthd: Adaptive efficient training for brain-inspired hyperdimensional computing. In *2019 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 1–4. IEEE, 2019.

[65] Mohsen Imani, Justin Morris, John Messerly, Helen Shu, Yaobang Deng, and Tajana Rosing. Bric: Locality-based encoding for energy-efficient brain-inspired hyperdimensional computing. In *Proceedings of the 56th Annual Design Automation Conference 2019*, page 52. ACM, 2019.

[66] Mohsen Imani, Tarek Nassar, Abbas Rahimi, and Tajana Rosing. Hdna: Energy-efficient dna sequencing using hyperdimensional computing. In *2018 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, pages 271–274. IEEE, 2018.

[67] Mohsen Imani, Abbas Rahimi, Deqian Kong, Tajana Rosing, and Jan M Rabaey. Exploring hyperdimensional associative memory. In *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 445–456. IEEE, 2017.

[68] Mohsen Imani, Sahand Salamat, Saransh Gupta, Jiani Huang, and Tajana Rosing. Fach: Fpga-based acceleration of hyperdimensional computing by reducing computational complexity. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, pages 493–498. ACM, 2019.

[69] Laurent Jacques, Jason N Laska, Petros T Boufounos, and Richard G Baraniuk. Robust 1-

bit compressive sensing via binary stable embeddings of sparse vectors. *IEEE Transactions on Information Theory*, 59(4):2082–2102, 2013.

[70] Michael Irwin Jordan, Terrence Joseph Sejnowski, and Tomaso A Poggio. *Graphical models: Foundations of neural computation*. MIT press, 2001.

[71] Daniel M Kane and Jelani Nelson. Sparser Johnson-Lindenstrauss transforms. *Journal of the ACM (JACM)*, 61(1):1–23, 2014.

[72] Pentti Kanerva. *Sparse distributed memory*. MIT press, 1988.

[73] Pentti Kanerva. The spatter code for encoding concepts at many levels. In *International Conference on Artificial Neural Networks*, pages 226–229. Springer, 1994.

[74] Pentti Kanerva. A family of binary spatter codes. In *International Conference on Artificial Neural Networks*, volume 1, pages 517–522, 1995.

[75] Pentti Kanerva. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation*, 1(2):139–159, 2009.

[76] Geethan Karunaratne, Manuel Le Gallo, Giovanni Cherubini, Luca Benini, Abbas Rahimi, and Abu Sebastian. In-memory hyperdimensional computing. *Nature Electronics*, pages 1–11, 2020.

[77] Behnam Khaleghi, Jaeyoung Kang, Hanyang Xu, Justin Morris, and Tajana Rosing. Generic: highly efficient learning engine on edge using hyperdimensional computing. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 1117–1122, 2022.

[78] Yeseong Kim, Mohsen Imani, Niema Moshiri, and Tajana Rosing. GenieHD: Efficient dna pattern matching accelerator using hyperdimensional computing. In *2020 Design, Automation & Test in Europe Conference & Exhibition*. IEEE, 2020.

[79] Yeseong Kim, Mohsen Imani, and Tajana Simunic Rosing. Efficient human activity recognition using hyperdimensional computing. In Krzysztof Janowicz, Werner Kuhn, Federica Cena, Armin Haller, and Kyriakos G. Vamvoudakis, editors, *Proceedings of the 8th International Conference on the Internet of Things, IOT 2018, Santa Barbara, CA, USA, October 15-18, 2018*, pages 38:1–38:6. ACM, 2018.

[80] George S Kimeldorf and Grace Wahba. A correspondence between bayesian estimation on stochastic processes and smoothing by splines. *The Annals of Mathematical Statistics*, 41(2):495–502, 1970.

[81] Denis Kleyko, Edward Paxon Frady, and Friedrich T Sommer. Cellular automata can reduce memory requirements of collective-state computing. *IEEE Transactions on Neural Networks and Learning Systems*, 33(6):2701–2713, 2021.

[82] Denis Kleyko, Dmitri A Rachkovskij, Evgeny Osipov, and Abbas Rahim. A survey on hyperdimensional computing aka vector symbolic architectures, part II: Applications, cognitive models, and challenges. *arXiv preprint arXiv:2112.15424*, 2021.

[83] Denis Kleyko, Dmitri A Rachkovskij, Evgeny Osipov, and Abbas Rahimi. A survey on hyperdimensional computing aka vector symbolic architectures, part I: Models and data transformations. *ACM Computing Surveys (CSUR)*, 2021.

[84] Denis Kleyko, Abbas Rahimi, Ross W Gayler, and Evgeny Osipov. Autoscaling Bloom filter: controlling trade-off between true and false positives. *Neural Computing and Applications*, 32:1–10, 2019.

[85] Denis Kleyko, Abbas Rahimi, Dmitri A Rachkovskij, Evgeny Osipov, and Jan M Rabaey. Classification and recall with binary hyperdimensional computing: Tradeoffs in choice of density and mapping characteristics. *IEEE Transactions on Neural Networks and Learning Systems*, 29(12):5880–5898, 2018.

[86] Brent Komer, Terrence C Stewart, Aaron Voelker, and Chris Eliasmith. A neural representation of continuous space using fractional binding. In *41st Annual Meeting of the Cognitive Science Society*, pages 2038–2043, 2019.

[87] Lukas Kull, Thomas Toifl, et al. A 3.1 mw 8b 1.2 gs/s single-channel asynchronous sar adc with alternate comparators for enhanced speed in 32 nm digital soi cmos. *IEEE Journal of Solid-State Circuits*, 48(12):3049–3058, 2013.

[88] Vitaly Kuznetsov, Mehryar Mohri, and Umar Syed. Multi-class deep boosting. *Advances in Neural Information Processing Systems*, 27, 2014.

[89] Mika Laiho, Jussi H Poikonen, Pentti Kanerva, and Eero Lehtonen. High-dimensional computing with sparse vectors. In *2015 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 1–4. IEEE, 2015.

[90] Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1988.

[91] Nicolas Y Masse, Glenn C Turner, and Gregory SXE Jefferis. Olfactory information processing in drosophila. *Current Biology*, 19(16):R700–R713, 2009.

[92] Colin McDiarmid et al. On the method of bounded differences. *Surveys in Combinatorics*, 141(1):148–188, 1989.

[93] Alisha Menon, Daniel Sun, Sarina Sabouri, Kyoungtae Lee, Melvin Aristio, Harrison Liew, and Jan M Rabaey. A highly energy-efficient hyperdimensional computing processor for biosignal classification. *IEEE Transactions on Biomedical Circuits and Systems*, 16(4):524–534, 2022.

[94] A Mitrokhin, P Sutor, C Fermüller, and Y Aloimonos. Learning sensorimotor control with neuromorphic sensors: Toward hyperdimensional active perception. *Science Robotics*, 4(30), 2019.

[95] Justin Morris, Kazim Ergun, Behnam Khaleghi, Mohsen Imani, Baris Aksanli, and Tajana Rosing. Hydrea: Towards more robust and efficient machine learning systems with hyperdimensional computing. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021.

[96] Fateme Rasti Najafabadi, Abbas Rahimi, Pentti Kanerva, and Jan M Rabaey. Hyperdimensional computing for text classification. In *Design, automation test in Europe conference exhibition (DATE), University Booth*, pages 1–1, 2016.

[97] Peer Neubert, Stefan Schubert, and Peter Protzel. An introduction to hyperdimensional computing for robotics. *KI-Künstliche Intelligenz*, 33(4):319–330, 2019.

[98] Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14, 2001.

[99] Ludwig Maximilian University of Munich: Software and Systems (SoSy) Lab. CPU energy meter. https://github.com/sosy-lab/cpu-energy-meter/, 2020.

[100] B.A. Olshausen and D.J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.

[101] Bruno A Olshausen and David J Field. Sparse coding of sensory inputs. *Current Opinion in Neurobiology*, 14(4):481–487, 2004.

[102] Anna Pagh, Rasmus Pagh, and S Srinivasa Rao. An optimal bloom filter replacement. In *Proceedings of the sixteenth annual ACM-SIAM Symposium on Discrete Algorithms*, pages 823–829, 2005.

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

[104] Yaniv Plan and Roman Vershynin. Dimension reduction by random hyperplane tessellations. *Discrete & Computational Geometry*, 51(2):438–461, 2014.

[105] T.A. Plate. *Holographic Reduced Representation: Distributed Representation for Cognitive Structures*. CSLI Lecture Notes (CSLI- CHUP) Series. CSLI Publications, 2003.

[106] Tony A Plate. Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6(3):623–641, 1995.

[107] DA Rachkovskij. Estimation of vectors similarity by their randomized binary projections. *Cybernetics and Systems Analysis*, 51:808–818, 2015.

[108] DA Rachkovskij. Formation of similarity-reflecting binary vectors with random binary projections. *Cybernetics and Systems Analysis*, 51(2):313–323, 2015.

[109] DA Rachkovskij, IS Misuno, and SV Slipchenko. Randomized projective methods for the construction of binary sparse vector representations. *Cybernetics and Systems Analysis*, 48:146–156, 2012.

[110] Dmitri A. Rachkovskij. Representation and processing of structures with binary sparse distributed codes. *IEEE Transactions on Knowledge and Data Engineering*, 13(2):261–276, 2001.

[111] Dmitriy A Rachkovskiy, Sergey V Slipchenko, Ernst M Kussul, and Tatyana N Baidyk. Sparse binary distributed encoding of scalars. *Journal of Automation and Information Sciences*, 37(6), 2005.

[112] Dmitriy A Rachkovskiy, Sergey V Slipchenko, Ivan S Misuno, Ernst M Kussul, and Tatyana N Baidyk. Sparse binary distributed encoding of numeric vectors. *Journal of Automation and Information Sciences*, 37(11), 2005.

[113] Maxim Raginsky and Svetlana Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *Advances in Neural Information Processing Systems*, pages 1509–1517, 2009.

[114] Abbas Rahimi, Simone Benatti, Pentti Kanerva, Luca Benini, and Jan M Rabaey. Hyperdimensional biosignal processing: A case study for emg-based hand gesture recognition. In *2016 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–8. IEEE, 2016.

[115] Abbas Rahimi, Sohum Datta, Denis Kleyko, Edward Paxon Frady, Bruno Olshausen, Pentti Kanerva, and Jan M Rabaey. High-dimensional computing as a nanoscalable paradigm. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 64(9):2508–2521, 2017.

[116] Abbas Rahimi, Pentti Kanerva, Luca Benini, and Jan M Rabaey. Efficient biosignal processing using hyperdimensional computing: Network templates for combined learning

and classification of ExG signals. *Proceedings of the IEEE*, 107(1):123–143, 2018.

[117] Abbas Rahimi, Pentti Kanerva, and Jan M Rabaey. A robust and energy-efficient classifier using brain-inspired hyperdimensional computing. In *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*, pages 64–69, 2016.

[118] Abbas Rahimi, Artiom Tchouprina, Pentti Kanerva, José del R Millán, and Jan M Rabaey. Hyperdimensional computing for blind and one-shot classification of eeg error-related potentials. *Mobile Networks and Applications*, 25:1–12, 2017.

[119] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing systems*, pages 1177–1184, 2008.

[120] Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. *Advances in neural information processing systems*, 21, 2008.

[121] Vikas Raunak, Vivek Gupta, and Florian Metze. Effective dimensionality reduction for word embeddings. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 235–243, 2019.

[122] F. Rosenblatt. The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[123] Mark Rudelson, Roman Vershynin, et al. Hanson-wright inequality and sub-gaussian concentration. *Electronic Communications in Probability*, 18, 2013.

[124] Walter Rudin. *Fourier analysis on groups*. John Wiley and Sons, Ltd, 1962.

[125] D.E. Rumelhart, J.L. McClelland, and the PDP Research Group. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*. MIT Press, 1986.

[126] Magnus Sahlgren. An introduction to random indexing. In *Methods and applications of semantic indexing workshop at the 7th international conference on terminology and knowledge engineering*, 2005.

[127] Sahand Salamat, Mohsen Imani, Behnam Khaleghi, and Tajana Rosing. F5-hd: Fast flexible fpga-based framework for refreshing hyperdimensional computing. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 53–62, 2019.

[128] Manuel Schmuck, Luca Benini, and Abbas Rahimi. Hardware optimizations of dense binary hyperdimensional computing: Rematerialization of hypervectors, binarized bundling,

and combinational associative memory. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 15(4):1–25, 2019.

[129] Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. A generalized representer theorem. In *International conference on computational learning theory*, pages 416–426. Springer, 2001.

[130] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *International conference on artificial neural networks*, pages 583–588. Springer, 1997.

[131] Clayton Scott. Ece 598 notes: Rademacher complexity of kernel classes. https://web.eecs.umich.edu/~cscott/past_courses/eecs598w14.

[132] Hajime Senuma. mmh3 python package. https://pypi.org/project/mmh3/, 2021.

[133] Ali Shafiee, Anirban Nag, Naveen Muralimanohar, et al. Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. *ACM SIGARCH Computer Architecture News*, 44(3):14–26, 2016.

[134] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[135] John Shawe-Taylor, Nello Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[136] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[137] Qinfeng Shi, James Petterson, Gideon Dror, John Langford, Alex Smola, and SVN Vishwanathan. Hash kernels for structured data. *Journal of Machine Learning Research*, 10(11), 2009.

[138] Paul Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial intelligence*, 46(1-2):159–216, 1990.

[139] Bharath Sriperumbudur and Zoltán Szabó. Optimal rates for random fourier features. *Advances in neural information processing systems*, 28, 2015.

[140] Ingo Steinwart. Consistency of support vector machines and other regularized kernel classifiers. *IEEE transactions on information theory*, 51(1):128–142, 2005.

[141] Dan D Stettler and Richard Axel. Representations of odor in the piriform cortex. *Neuron*, 63(6):854–864, 2009.

[142] Aaron Stillmaker and Bevan Baas. Scaling equations for the accurate prediction of cmos device performance from 180 nm to 7 nm. *Integration*, 58:74–81, 2017.

[143] Anthony Thomas, Sanjoy Dasgupta, and Tajana Rosing. A theoretical perspective on hyperdimensional computing. *Journal of Artificial Intelligence Research*, 72:215–249, 2021.

[144] Anthony Thomas, Sanjoy Dasgupta, and Tajana Rosing. A theoretical perspective on hyperdimensional computing. *Journal of Artificial Intelligence Research*, 72:215–249, 2021.

[145] Anthony Thomas, Behnam Khaleghi, Gopi Krishna Jha, Nageen Himayat, Ravi Iyer, Nilesh Jain, and Tajana Rosing. Streaming encoding algorithms for scalable hyperdimensional computing. *arXiv preprint arXiv:2209.09868*, 2022.

[146] Glenn C Turner, Maxim Bazhenov, and Gilles Laurent. Olfactory representations by drosophila mushroom body neurons. *Journal of Neurophysiology*, 99(2):734–746, 2008.

[147] Salil P Vadhan et al. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1–3):1–336, 2012.

[148] Vladimir Vapnik. The support vector method of function estimation. In *Nonlinear Modeling*, pages 55–85. Springer, 1998.

[149] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1999.

[150] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

[151] Martin J Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*, volume 48. Cambridge University Press, 2019.

[152] Dominic Widdows and Trevor Cohen. Reasoning with vectors: A continuous model for fast robust inference. *Logic Journal of the IGPL*, 23(2):141–173, 2015.

[153] Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. *Advances in neural information processing systems*, 13, 2000.

[154] Rachel I Wilson. Early olfactory processing in drosophila: mechanisms and principles. *Annual Review of Neuroscience*, 36:217–241, 2013.

[155] Tao Yu, Yichi Zhang, Zhiru Zhang, and Christopher De Sa. Understanding hyperdimensional computing for parallel single-pass learning. *arXiv preprint arXiv:2202.04805*,

2022.

[156] Jian Zhang, Avner May, Tri Dao, and Christopher Ré. Low-precision random fourier features for memory-constrained kernel approximation. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1264–1274. PMLR, 2019.

[157] Jinjie Zhang and Rayan Saab. Faster binary embeddings for preserving euclidean distances. *arXiv preprint arXiv:2010.00712*, 2020.

[158] Dengyong Zhou and Bernhard Schölkopf. Learning from labeled and unlabeled data using random walks. In *Pattern Recognition: 26th DAGM Symposium, Tübingen, Germany, August 30-September 1, 2004. Proceedings 26*, pages 237–244. Springer, 2004.

[159] Zhuowen Zou, Yeseong Kim, Farhad Imani, Haleh Alimohamadi, Rosario Cammarota, and Mohsen Imani. Scalable edge-based hyperdimensional learning system with brain-like neural adaptation. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15, 2021.