The Dissertation Committee for Yong Seok Yoo
certifies that this is the approved version of the following dissertation:

Multi-scale error-correcting codes and their decoding using belief propagation

Committee:

Ila Fiete, Supervisor

Sriram Vishwanath, Supervisor

Alan Bovik

Ahmed Tewfik

Jonathan Pillow

Multi-scale error-correcting codes and their decoding using belief propagation

by

Yong Seok Yoo, B.S.E.E.; M.S.E.E.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2014

"I know that you can do all things; no purpose of yours can be thwarted. You asked, 'Who is this that obscures my plans without knowledge?' Surely I spoke of things I did not understand, things too wonderful for me to know. You said, 'Listen now, and I will speak; I will question you, and you shall answer me.' My ears had heard of you but now my eyes have seen you. Therefore I despise myself and repent in dust and ashes" (Job 42:2-6).

Acknowledgments

I wish to thank numerous people. First, I am grateful to academic advisers, Professors Ila Fiete and Sriram Vishwanath. I am fortunate to have excellent academic advisers from two fascinating fields, computational neuroscience and information theory. Professor Fiete taught me critical thinking skills and basics of research. Professor Vishwanath demonstrated how to enthusiastically explore new ideas. I am indebted to Professor Jonathan Pillow for thought-provoking discussions. I am also grateful to Professors Alan Bovik and Ahmed Tewfik for insightful comments and feedbacks. I would like to thank former and current members of Fiete lab and LINC group, especially Ozan, Kijung, Birgit, Harold, Muryong, Sang Hyun, and Abhik for discussions and helps. Dear friends in Austin, to name just few among them, Insoo, Juhun, Youngchun, Dohyung, Namyoon, Jiho, Duhee, Joonhyun, and Dongoh have been valuable source of ideas, supports, and also fun. Most of all, I would like to express my deepest gratitude and love to Grandmother, Father, and Mother. Their sacrificial love and encouragement have always been there for me. My sister's family, especially lovely nephew Sunghoon, has been a blessing to my humble life in Austin. I would like to thank my uncles Jinkeun, the first Longhorn in my family, and Youngchul for wisdom and supports. Last but not least, I am deeply thankful to my beautiful wife Woori. She is a godsend; her unconditional love and timely advices have been essential for completing my doctoral study.

Multi-scale error-correcting codes and their decoding using belief propagation

Publication No. _____

Yong Seok Yoo, Ph.D.
The University of Texas at Austin, 2014

Supervisors: Ila Fiete
Sriram Vishwanath

This work is motivated from error-correcting codes in the brain. To counteract the effect of representation noise, a large number of neurons participate in encoding even low-dimensional variables. In many brain areas, the mean firing rates of neurons as a function of represented variable, called the tuning curve, have unimodal shape centered at different values, defining a unary code. This dissertation focuses on a new type of neural code where neurons have periodic tuning curves, with a diversity of periods. Neurons that exhibit this tuning are grid cells of the entorhinal cortex, which represent self-location in two-dimensional space. First, we investigate mutual information between such multi-scale codes and the coded variable as a function of tuning curve width. For decoding, we consider maximum likelihood (ML) and plausible neural network (NN) based models. For unary neural codes, Fisher information increases with narrower tuning, regardless of the decoding method.

By contrast, for the multi-scale neural code, the optimal tuning curve width depends on the decoding method. While narrow tuning is optimal for ML decoding, a finite width, matched to statistics of the noise, is optimal with a NN decoder. This finding may explain why actual neural tuning curves have relatively wide tuning. Next, motivated by the observation that multi-scale codes involve non-trivial decoding, we examine a decoding algorithm based on belief propagation (BP) because BP promises certain gains in decoding efficiency. The decoding problem is first formulated as a subset selection problem on a graph and then approximately solved by BP. Even though the graph has many cycles, BP converges to a fixed point after few iterations. The mean square error of BP approaches to that of ML at high signal-to-noise ratios. Finally, using the multi-scale code, we propose a joint source-channel coding scheme that allows separate senders to transmit complementary information over additive Gaussian noise channels without cooperation. The receiver decodes one sender's codeword using the other as side information and achieves a lower distortion using the same number of transmissions. The proposed scheme offers a new framework to design distributed joint source-channel codes for continuous variables.

Table of Contents

List of Figures

Chapter 1

Introduction

The introduction previews a connection between neuroscience and engineering from coding perspective. We first review error-correcting codes in communication systems. The same notion of error correction by redundancy is observed in the brain, which motivates this work.

1.1 Error-correcting codes for reliable communication over noisy channels

Researchers have investigated error-correcting codes to achieve reliable communication over noisy channels. The sender wishes to communicate a message to the receiver through a noisy channel. In order to reduce the effect of the noise, the message is encoded to a codeword in a redundant manner and this codeword is transmitted through the noisy channel. The redundancy in the codeword enables the receiver to recover the message while minimizing the effect of the noise.

Since Shannon's celebrated work of proving the existence of random codes that achieve an arbitrary small error probability with the minimum amount of redundancy [80], there have been persistent efforts to find practical

error-correcting codes. Among those, linear codes have enjoyed great success due to their analytical tractability, easier design, and high data rate. Examples of widely used linear codes include Reed-Solomon (RS) [74] and Bose-Chaudhuri-Hocquenghem (BCH) [14, 15, 51] codes. However, high decoding complexity limits applications of linear codes. For example, Berlekamp *et al.* show that decoding a linear code is NP-complete [11]. Readers are referred to [58] for examples of decoding various linear codes. Gallager's low-density parity-check (LDPC) codes [41] greatly reduces decoding complexity. LDPC codes are efficiently decoded with local computations on a graph using iterative algorithms [75]. Low decoding complexity and high data rate of LDPC codes have become integral parts of the research on linear codes. The idea of iterative decoding is used in the second part of the dissertation to design a decoding algorithm with low computational complexity.

Now, attention is increasingly focused on nonlinear codes. Unusual constructions of nonlinear codes and advantages over linear codes have gained a lot of interests [5, 47]. The dissertation is concerned with a family of nonlinear codes called shift-map codes. With shift-map codes [25, 92, 94], a continuous message is represented by the continuous codeword with a longer length. Thus, redundancy is added by a mapping from lower dimensional to higher dimensional spaces. The shift-map generates the codeword from a continuous message by multiplications followed by modulo operations. Consequently, the message is encoded into its residues with respect to distinct moduli. In this sense, shift-map codes are closely related to their integer counterpart

redundant-residual-number-system (RRNS) codes [84]. This research is motivated by an intriguing connection between shit-map codes and the brain's multi-scale code for representing self-location [66], discussed in the following section.

1.2 Motivation from the brain's error-correcting codes for self-location

In the brain, networks of neurons collectively process information, but individual neurons are noisy [22, 33, 35, 52, 85, 86]. Neurons interact with one another via stochastic opening and closing of ion channels [50] and the release of neurotransmitters [22, 33]. Consequently, a single neuron's response, commonly measured by either membrane potential or firing rate, varies from trial to trial even for the same input [52, 85, 86]. Then, how can the brain perform any reliable information processing with such noisy components?

Representing one variable with a large number of neurons, hence redundancy, reduces the effect of the inherent neural variability. In many brain areas, the mean firing rates of neurons vary as a function of the represented variable and are called tuning curves. Neurons in the sensory and the motor cortices have uni-modal tuning curves with different centers and this redundant and unary representation defines the *population code* [32, 44, 70, 104].

Figure 1.1 illustrates an example of population codes observed in monkey's visual cortex. Figure 1.1A shows a visual stimulus, a bar with a orientation θ. Firing rates of neurons in the primary visual cortex are modulated by

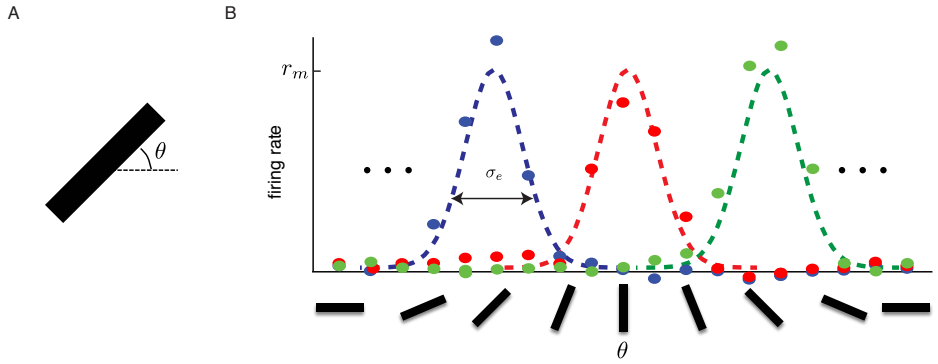

Figure 1.1: An example of population codes in the primary visual cortex. A shows a visual stimulus with orientation θ. In B, dashed curves with different colors represent individual neurons' tuning curves as a function of θ and circles noisy firing rates at a given time.

the orientation or the bar [53]. In Figure 1.1B, dashed curves with different colors represent tuning curves of three neurons as function of the bar orientation θ. Circles indicate firing rates at a given time, which deviates from the tuning curves according to the Poisson distribution.

Fisher information (FI) quantifies information about input variable from a population code and the dependence of FI on tuning curve parameters has been studied [10, 32, 65, 69, 72, 77, 105]. By Cramer-Rao bound, the inverse of FI determines the minimum mean square error that any unbiased estimator can achieve [31]. Thus, lager FI implies a lower estimation error with an ideal estimator. One of the important parameters of population codes is the number of neurons (N) participating in encoding, which determines the amount of redundancy. Under the assumption of uniformly distributed tuning curve centers over the input domain, FI linearly scales with N [32]. Thus, the

larger N is, the lower the estimation error is; increased redundancy leads to lower sensitivity to errors.

The FI also depends on tuning curve shape. Increasing the maximum firing rates (r_{max} in Figure 1.1B) of all neurons also increases FI [32]. This could be understand as stronger noise reduction by increasing signal power. The tuning curve width (σ_e in Figure 1.1B) controls the sparsity of the code. If σ_e is large, more neurons are active at a given time. On the other hand, if σ_e is small, only few neurons are active. FI depends on σ_e in a dimension dependent manner. When the encoded variable is low-dimensional, narrower tuning increases FI. By contrast, when the encoded variable is high-dimensional, wider tuning increases FI. The intuition for this difference is that for low-dimensional input, it is better to have only few neurons with high sensitivity. Increasing σ_e sacrifices sensitivity but increases the number of active neurons from which information is extracted. The latter dominates the former in a high-dimensional space.

It remains an active area of research understanding how information is processed in higher level structures of the brain [49, 56, 57, 59, 73, 98]. Among those structures, the hippocampus (HPC) and the entorhinal cortex (EC) have attracted much attention due to their critical roles in learning and memory [4]. The EC is an interface between the HPC and the neocortex and both the EC and the HPC are involved in processing self-location [66]. The HPC represents self-location in a way similar to classical population codes [67, 98]. Recently, multi-scale spatial representation has been discovered in the EC

[48, 66]. The unique nature of such multi-scale representation in the EC offers an opportunity to understand coding principles in deeper areas of the brain.

In the EC, multiple modules of grid cells (GCs) with different scales represent self-location [48,90]. GCs in each module have periodic tuning curves as a function of animal location. Thus, GCs in each module form a population code encoding a *phase* of the location. There are multiple modules with distinct periods [48, 90]. Thus, GC modules encode the self-location as *a set of phases* with different periodicities. This neural codes in the EC share the following two principles with classical population codes. First, a large number of neurons in each module encode phase; redundancy. Second, only a fraction of neurons are active at a given time; sparsity. In addition, the additional structure across modules improves error correcting capabilities [36, 60, 89]. However, this gain comes at the cost of occasional large errors and difficulty in decoding. Thus, we study the dependence of coding accuracy on tuning curve width in multi-scale multi-population codes, and compare with results from single-population codes with unimodal tuning curves.

1.3 Summary of contributions

The three main contributions are summarized as follows.

First, mutual information between multi-scale neural codes and a low-dimensional input variable is quantified as a function of tuning curve width. As reviewed in Section 1.2, the Fisher information of unary neural codes increases with narrower tuning for a low-dimensional input variables. To under-

stand multi-scale neural codes, we us mutual information, instead of Fisher information, as information measure and consider maximum likelihood (ML) and plausible neural network (NN) based models. In contrast to unary neural codes, the optimal tuning curve width depends on decoding method for the multi-scale neural code. While narrow tuning is optimal for ML decoding, a finite width, which is matched to statistics of the noise in the problem, is optimal with a NN decoder. This finding is discussed in relation to relatively wide tuning in the brain.

Next, we explore a decoding algorithm for multi-scale codes based on belief propagation (BP). The decoding problem is first formulated as a subset selection problem on a graph and then solved by a BP algorithm. Even though this graph has many cycles, the proposed algorithm converges to a fixed point within few iterations. Convergence and accuracy of the proposed algorithm are investigated in light of previous studies on BP on cyclic graphs. Numerical simulations show that the mean square error of BP approaches to that of ML for high signal-to-noise ratios.

Finally, the multi-scale code is applied to a distributed coding problem. We propose a joint source-channel coding scheme which allows separate senders to transmit complementary information over additive Gaussian noise channels without cooperation. For this purpose, the shift-map code is generalized to a new family of code similar to the multi-scale representation of the brain. The receiver decodes one sender's codeword using the other as side information and achieves a lower distortion using the same number of transmissions. The

proposed scheme is interpreted as an extension of binning discrete codebook, a widely used technique for multi-user channels. Thus, the proposed scheme offers a new method to structure a continuous codebook for distributed joint source-channel codes.

1.4 Organization

The rest of the dissertation is organized as follows. Chapter 2 addresses the mutual information between GCs and self-location as a function of tuning curve width. Chapter 3 covers a decoding algorithm for multi-scale codes using belief propagation. In Chapter 4, the multi-scale neural code is applied to design a joint source-channel code for distributed coding. Chapter 5 includes conclusion and suggestions for future works.

Chapter 2

Multi-scale error-correcting codes in the brain

2.1 Introduction

The inherent variability in neural activity – including stochastic ion channel dynamics [50], stochastic vesicle release [91], and variable spike output for repeated stimuli [78, 85, 86] – means that inferring the value of an encoded variable from neuron responses carries with it a degree of uncertainty.

If the variable is encoded in the responses of a large number of neurons, the estimation error can be reduced [44, 72, 77, 105]. In such *population codes*, the resulting estimation error depends on many parameters, including the number of neurons (N), the shape and width of the neural tuning curves, the neural nonlinearity, and the model for neural variability. The parametric scaling of estimation error for several population codes has been derived by computing the Fisher information, whose inverse specifies the minimum variance of any unbiased estimator [10, 32, 65, 69, 72, 77, 105].

The population codes found in the sensory and motor peripheries, and even some cognitive codes, typically consist of ramp-like or unimodal (single) bump-like responses per neuron, as a function of the encoded variable [23, 24, 44, 53, 55, 93, 98, 104]. Different neurons in the population are often

described by simple shifts of a canonical tuning curve. Assuming that neural spikes are generated by an inhomogeneous point process (e.g. Poisson process), with spike rates given by the underlying neural tuning curves, the Cramer-Rao asymptotic bound on estimation error, given by the inverse of the Fisher information, can decrease with neuron number as $\sim 1/N$ [10, 32, 72, 105] (but the actual squared error, which may be considerably larger than the bound, might not decrease as fast as $\sim 1/N$ [10]). Common forms of correlations between neurons can affect the estimation error, but do not generally improve the basic scaling with N [2, 79, 87].

Besides neuron number, estimation accuracy also depends on tuning curve width. For unimodal (single-bump) codes for a non-periodic variable, the scaling with tuning curve width is strongly dependent on the dimension of the coded variable [72, 105]. For one dimension, the narrower the tuning curves, the better; for two dimensions, the Fisher information is independent of tuning curve width. For periodic neural codes, narrower tuning curves are better, in both one and two dimensions [65]. If the represented quantity is a latent variable that is never directly observed by the population, but drives the population state through a noisy process (for instance, photon noise, noisy sensors, etc.), then the information in the population code about the represented quantity saturates at the level of this noisy process. In the presence of such "sensing" noise, narrower tuning curves are better up to the saturation point, and then the gains level off: the Fisher information and mutual information do not improve. These results hold both for ideal (maximum likelihood,

referred to as ML in the rest of this paper) decoders and for simpler decoders including the population vector and neural network decoders [77].

Multi-scale population codes encode a non-periodic variable as a set of phases, each computed with respect to a distinct periodic response. A motivating example for the present work is the representation of 2D spatial location in mammals by several populations of grid cells [36, 48, 90]. Neurons in one population display a common periodic response with offset preferred phases, and the set of populations supply a set of distinct periods. In this work, we will consider the case where the largest period is significantly smaller than the range of the non-periodic variable [36]. Thus, even in the absence of noise, each population only represents partial information about the encoded variable. The variable can only be estimated unambiguously over its range by considering most or all of the populations.

Multi-scale population codes differ strikingly from unimodal or ramp-like population codes commonly seen in the sensory and motor peripheries, in at least one respect: the dynamic range of multi-scale population codes, or equivalently the range divided by the resolution of the decoded variable, grows exponentially with N [61, 89], in contrast with the polynomial growth with N seen in the unimodal or ramp-like population codes. This exponential scaling of the decoded dynamic range is possible even when the spatial periods do not exhibit an exponential range of scales ([36,89], in contrast to the exponentially large range of scales in the periods assumed in [61]). Here, we study the dependence of coding accuracy on tuning curve width in multiperiodic multi-

population codes, and compare with results from single-population codes.

Throughout this work, we will consider the represented quantity to be a latent variable. Because the populations are distinct, and possibly independently sample or sense the latent variable, we will consider the effect of independent "sensing" noise in each population's representation of the latent variable.

We show that with ideal observer decoding through maximum likelihood, the results for optimal multi-population tuning width closely match those for classical population codes, even in the presence of independent sensing noise. However, with neural network decoding, we show that a finite encoding tuning curve width is optimal, in direct contrast with results in unimodal or ramp-like population codes.

2.2 Background and Methods

2.2.1 Description of multi-period population coding

The motivation for considering a set of different populations codes characterized by different periodic responses is the representation of spatial location by grid cells [48]. As an animal explores the floor of a 2D enclosure, a single grid cell fires at multiple locations, at the vertices of a virtual equilateral triangular lattice that tiles the floor, Figure 2.1A. Nearby cells share a common response period and orientation, and differ only in phase – i.e., through rigid spatial translations of the periodic spatial pattern. We define a group of such cells as *one population* or module [90]. The networks appear to

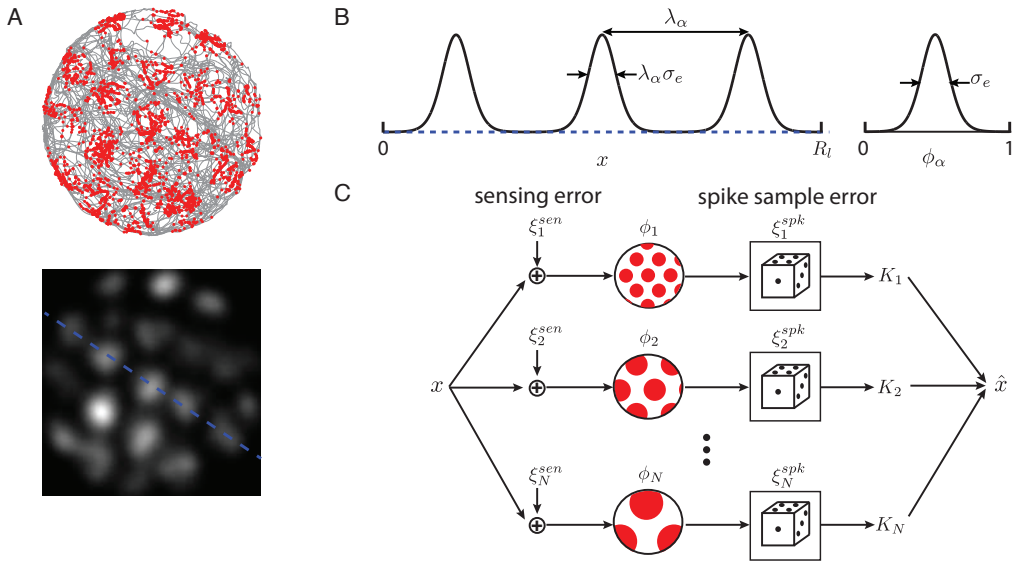

Figure 2.1: Multi-scale multipopulation codes: motivation from grid cells. (A) A grid cell in the medial entorhinal cortex fires whenever the animal visits a set of periodically arranged locations, that form the vertices of a (virtual) equilateral triangular lattice. Top plot: gray line, trajectory of the animal as it foraged on the floor of a square box; red dots, locations of the spikes emitted by one grid cell (data from [1]). Bottom plot: smoothed spatial firing rate obtained from the plot above. Brighter values correspond to higher firing rates. Dashed blue line is a 1 dimensional slice that is considered in B. (B) The firing rate response may be viewed as a periodic multi-peaked function of space (B, left) or as a unimodal (single-bump) function of phase (B, right). The tuning curve width σ_e is defined as the ratio of the standard deviation of the activity bumps to the period of the spatial response. (C) The external variable x (animal location) is a latent variable that drives the grid populations. The populations, indexed by $n = 1, 2, \cdots, N$, encode partial information about x through their phases ϕ_n, with respect to their spatial response periods λ_n. Red filled circles within an black open circle represent the instantaneous population response of one group of cells; the population pattern phase is a function of the variable x. The population periods increase systematically with the index n, corresponding to the displacement of the population toward the ventral border of the dorsoventral axis of the entorhinal cortex, along which grid cells are found [48].

be disjoint, with discrete jumps in period across networks [90], as predicted in [19, 36, 39, 64]. The locations of the spatially periodic response peaks of all cells in one population together encode location as only a spatial phase with respect to their shared period and orientation [36].

Let $n = 1, \cdots, N$ be the population index, and let each population contain M neurons. Let $\vec{x}$ be the latent represented variable. We assume that the ith cell in the nth population emits spikes according to an inhomogeneous point process, with a time-varying firing rate given by:

$$r_{ni}(\vec{x}, t) = f(|\vec{\phi}_n(\vec{x}) - \vec{\phi}_i|). \tag{2.1}$$

Here, $\vec{\phi}_n(\vec{x})$ is the population activity phase, which depends on the represented variable $\vec{x}$. Each coordinate of the phase is a circular variable, normalized to lie in the circular interval $[0, 1)$, where 0 is close to 1. The preferred phase of the ith neuron in a populaton is designated $\vec{\phi}_i$. f is some unimodal bump function in the periodic unit cell, for example, a circular Gaussian. If the spiking process is Poisson, as we will assume here, then the n, ith cell emits K_{ni} spikes in a time interval $[t, t + dt)$ with probability

$$P(K_{ni}|r(\vec{x})) = \frac{(r_{ni}dt)^{K_{ni}} e^{-r_{ni}dt}}{K_{ni}!}, \quad n = 1, 2, \ldots, N, \quad i = 1, 2, \ldots M. \tag{2.2}$$

The variability in spiking means that any estimate $\hat{\vec{\phi}}_n$ of the network phase $\vec{\phi}_n(\vec{x})$ from the network spikes is inevitably noisy. The estimation noise ξ_n^{spk} is defined by:

$$\hat{\phi}_n(\vec{x}) = \phi_n(\vec{x}) + \xi_n^{spk}. \tag{2.3}$$

Henceforth, we will refer to this noise as the *spike sample error*, Figure 1B, because it refers to the error inherent in decoding the network phase $\vec{\phi}_n(\vec{x})$ from a finite sample of noisy spikes (over an interval dt). With an unbiased optimal estimator and a large number of neurons in each grid network (large M), the spike sample error is Gaussian, with zero mean and variance lower-bounded by the inverse Fisher information [69, 77, 105].

Next, any value of the represented variable $\vec{x}$ can be mapped to a vector phase $\vec{\bar{\phi}}_n(\vec{x})$ in a periodic grid, with respect to the period and geometry of the grid. Thus, if the encoded population phases $\vec{\phi}_n(\vec{x})$ exactly represent $\vec{x}$ up to a periodic transformation, we will have:

$$\vec{\phi}_n(x) = \vec{\bar{\phi}}_n(x). \tag{2.4}$$

However, the populations likely do not have direct access to $\vec{x}$. For instance, in the example of grid cells, if each grid network integrates self-motion cues to estimate its spatial phase and there is a very small noise in either the self-motion cues or in the integration process [16, 20, 28, 99], that population's phase will grow increasingly mismatched from the phase corresponding to the true location over time [19]. Even if the grid cell system's motion-derived location estimate is corrected by a decoder that exploits the inherent error-correcting properties of the grid code [89] or by the help of external sensory cues [21, 97], errors will accumulate in the periods between subsequent corrections. Generally speaking, neurons can only obtain and encode an *estimate* of the true value of an external variable in the world, with the estimate subject

to errors from the sensory system and other sources. Therefore, we include the possibility of error in the spatial phase encoded by each network. These errors are assumed to be independent across networks and coherent within a network; we model these errors ξ_n^{sen} as uncorrelated Gaussians, with zero mean and standard deviation σ^{sen}:

$$\phi_n(\vec{x}, t) = \left(\bar{\phi}_n(\vec{x}) + \xi_n^{sen,k}(t)\right) \bmod 1. \tag{2.5}$$

As above, the bar on the kth phase component $\bar{\phi}_n^k$ refers to the error-free phase for the variable $\vec{x}$.

Henceforth, we refer to the phase noise ξ_n^{sen} as the *sensing error*, because it represents all errors that intervene between the actual value of the variable in the external world and value that the network will represent, Figure 1B.

To summarize, relative to the actual value of the external variable $\vec{x}$, the final decoded estimate of $\vec{x}$ from the activity of neurons will include the effects of sensory noise, ξ_n^{sen}, as well as the effects of spike sample noise, ξ_n^{spk}.

2.2.2 Tuning curve width

We will assume that across populations, the widths of the activity peaks scale in proportion to the population response periods, so that the ratio of activity peak width to period remains constant across populations. This assumption is motivated by the grid cell data [48]. An interpretation of the finding of equal tuning curve widths is that, if the number of neurons in each grid network is the same, then the error in decoding the encoded phase in

each population from its spikes is the same across populations. We refer to the constant ratio of activity peak width to period across populations as the *tuning curve width*, σ_e, of the multi-population code (Figure 1C-D).

2.2.3 Maximum likelihood and neural network decoders

We study the dependence of estimation error on tuning curve width for two decoders, first, an ideal decoder that estimates the value of the encoded variable that maximizes the likelihood of the data (ML decoding), and second, a plausible neural network (NN) decoder. The NN decoder is constructed as a set of weighted sums of all grid cell outputs, with a max operation performed to select the biggest output (see Appendix A.1 for details). The sum and max operations we consider may be envisioned as being performed by the feedforward projections to a decoding layer of neurons, with winner-take-all dynamics through global inhibition in the decoding layer, respectively.

2.3 Results

2.3.1 A geometric view of neural coding.

When a relatively low-dimensional variable is coded by a higher-dimensional representation – for instance, in the activities of a large number of neurons – we may view the code as a low-dimensional embedding of a set of points (if the variable is continuous, then a manifold of points) in the high-dimensional space. A schematic view of a 1D variable coded by embedding in a 2D representational space is given in Figure 2.2.

In general, the advantage of embedding a variable in a higher dimensional space is the potential for error reduction. When the (high-dimensional) representation is perturbed by (high-dimensional) noise, the system state is bumped off the coding manifold; however, the perturbed representation may be decoded by mapping it back to the coding manifold. The high-dimensional noise is now largely erased, except for the projection of the noise along the low-dimensional coding line. Thus, for a D-dimensional variable embedded in an N-dimensional representational space, the residual squared error that results from a unit-length high-dimensional perturbation scales as D/N – a significant improvement when $D \ll N$.

However, such embeddings can also result in large, non-local errors. When the perturbation is sufficiently large, a point on the coding manifold may end up closer to a remote segment of the manifold. When decoded by mapping onto the closest point on the manifold, the result will be a large non-local error [10, 81, 89], also known as a *threshold error* [81]. The probability of threshold error depends on the nature of the perturbation and on the separation of segments of the coding line from one another.

If the dimensions (N and D, respectively) and ranges of the encoded variable and the representational space are held fixed, then the stretch factor L of the coding manifold embedded in the representational space can be independently varied, to be larger or smaller. The larger the stretch factor, the smaller the residual local error for a given high-dimensional perturbation, as a fraction of the total range of the coded variable. However, when a more

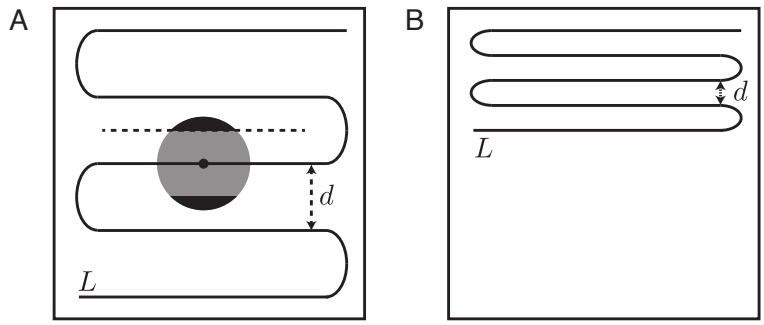

Figure 2.2: Geometric view of analog population codes: local versus threshold errors. Neural population codes may be viewed as the embedding of a low-dimensional coding manifold in the high-dimensional state space of neural activity. (A) Consider a 1D variable in the unit interval, represented by a 1D coding line in a higher-dimensional coding space. Suppose the length of the coding line is L, and suppose the curve bends back toward itself, with a minimum separation d between segments of the embedded curve (norm computed in the full-dimensional coding space). If noise-driven perturbations are small enough that projecting the perturbed state to the nearest segment (with *nearest* corresponding to the maximum likelihood of the perturbed point across points on the coding manifold based on the noise model, so we may view the norm as the inverse likelihood) based on likelihood of the coding line restores the system to the correct coding line segment, then the error is a *local* error. For a given noise, the magnitude of local errors scales as $1/L$: the longer the coding line, the smaller the error. However, perturbations that exceed $d/2$ in size, when mapped to the closest coding line, can fall on an adjacent segment that corresponds to a remote value of the coded variable. These *threshold* errors occur with higher frequency for a given noise when the coding line is longer, because d (weakly) shrinks as L increases. They will also occur with higher frequency if the coding line is poorly spaced relative to the noise model (B) or if the decoder does not take into account the correct noise model.

stretched coding manifold is packed into the same representational space, the segments are necessarily closer together (minimum spacing quantified by separation d), and can lead to an increase in threshold error. The increase in

threshold error probability typically grows weakly with L, if the coding manifold is much lower dimensional than the representational space, and if the structure of the embedding does not deteriorate with L – i.e., the embedding is still well-spaced within the coding space. At the same time, the structure and thus the separation properties of an embedding can also be varied, even for fixed L. For example, concentrating the coding manifold in one small subregion of the representational space and wasting the rest of the space can result in lower separation between coding segments, and an increase in threshold error probability, Figure 2.2B.

Because we are interested in neural population codes and their parameters, we seek to understand how the stretch factor L, the code structure, the separation d and the decoding error vary as a function of tuning curve width. A geometric view reveals global and qualitative properties of a code, augmenting approaches rooted in Fisher information, which provides strictly local information and because it ignores the probability of threshold error, is a sometimes brittle estimator of expected error and information [10, 13].

FI does not capture the effects of threshold errors, which can have a large influence on squared error and Mutual information (MI). For this reason, we will also compute MI in this work. In what follows, we study how the tuning curve width in neural codes affects both the FI and MI. We will consider information from an ideal observer perspective (ML decoding) and from the perspective of more plausible neural network decoders.

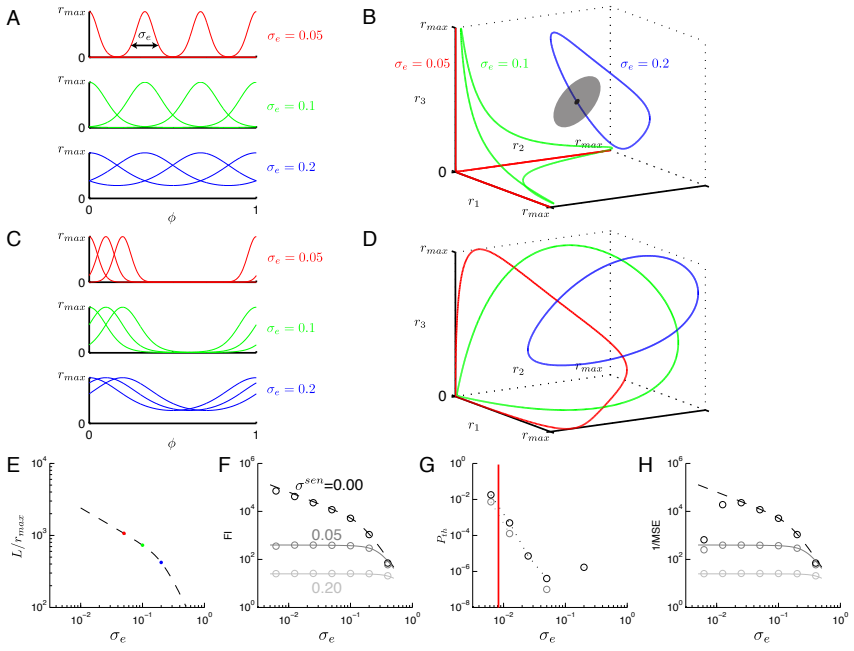

Figure 2.3: A review and geometric view of single-population codes. (A,C) The circular normal tuning curves of three cells from a much larger population encoding a circular variable. The tuning width is given by σ_e, and all curves are translations of one another. (B,D) The firing rate vector traces out a curve or coding line in the 3-dimensional space, which is itself a small subspace of the full coding space of all neurons. Black dot: a noise-free response; gray ball: schematic of the neighborhood of coding space around the black dot that is visited when neural spiking is Poisson. (E) The length L of the coding line increases as σ_e decreases. Colored dots indicated the specific values of σ_e from (A-D). (F) Fisher information (FI) in the absence of sensing error (dashed, $\sigma_{sen} = 0$), and when the sensing error is non-zero (solid; $\sigma_{sen} = 0.05$). Solid and dashed lines show analytically calculated FI for single networks (A.52 in Appendix A.3). Circles indicate numerically estimated FI, by measuring the inverse of the variance of the central peak in the posterior. (G) Threshold error varies very weakly with σ_e and is zero for most of the range. However, when the number of active neurons in a bump decreases to order 1, it sharply increases. (H) Mean square errors (circles) compared with the predictions from FI (curves). Parameters: $N = 256$, $r_{max}\Delta t = 1$, and 10^7 samples.

2.3.2 Single-population codes: a review and a geometric view

If N neurons with peak firing rate r_m represent a scalar variable with identical single-bump tuning (the tuning curves of different neurons are shifted copies of one another, Figure 2.3A) the representational space is N-dimensional and the coding manifold is a (1D) line, Figure 2.3A. For a circular variable, the embedded line is a ring, Figure 2.3B.

Single-population codes have been extensively analyzed [10, 72, 77, 105]. For a fixed number of neurons M, FI about an one-dimensional encoded variable ϕ increases for narrower tuning.

$$J(\phi) \quad = \quad \left((2\pi)^2 M r_m \Delta t\right) \kappa_e \left(\frac{I_1(\kappa_e)}{e^{\kappa_e}}\right), \tag{2.6}$$

where r_m is the maximum firing rate, $\kappa_e = 1/(2\pi\sigma_e)^2$, and $I_1(\kappa_e)$ is the first order modified Bessel function of the first kind which scales as e^{κ_e} for a large κ_e (Appendix A.3). Thus, $J(\phi)$ increases with decreasing σ_e. From a geometric viewpoint, this corresponds to the fact that as tuning curves become narrower, the length or stretch factor L of the coding line increases, Figure 2.3A-D (also see Appendix A.2). Thus, a unit of high-dimensional noise produces diminishing local errors with narrower tuning curves, Figure 2.3E.

At very narrow tuning, when order 1 neurons are active at any given time, it is possible for another set of neurons to become, by chance, more active than the neurons in the active bump (if their baseline firing rates are non-zero), or for neurons in the active bump to become inactive, leading to threshold error ([10] and D. Schawab and I. Fiete, unpublished work), Figure

2.3G. For broad tuning, many active neurons signal the neighborhood of the encoded variable, thus all errors are local.

Throughout this work, for single and multi-population codes, we will remain within the *population coding* regime, away from the regime where only one or a few neurons are active at a given time and, and therefore, away from the regime of threshold error in single population codes. We will consider regimes for the tuning curve width σ_e in which $\sigma_e N$, the number of active neurons, is always much greater than 1, so that the probability of threshold error in single populations is effectively zero. The smallest value of σ_e used in the rest of the paper is 0.0125, which corresponds to a threshold error probability of $< 2 \times 10^{-7}$ in the single populations (See Appendix A.5).

2.3.3 Tuning curve widths for latent variable representation and sensing noise

Suppose the variable does not directly drive the neural representation, and instead drives a sensory system that then drives the representation. We model this case by inserting a Gaussian noise term between the (now latent) variable and the neural representation. We call this the *sensing noise*. When the widths of the tuning curves in the population code are decreased to below the standard deviation of the sensing noise, the growth in FI in the code about the latent variable saturates, and there is no further increase for narrower tuning, Figure 2.3F. This happens because the inverse FI cannot exceed $(\sigma^{sen})^2$, regardless of tuning curve width.

2.3.4 Multiperiodic multi-population codes: with ML decoding narrower tuning curves are optimal until saturation due to sensing noise.

When sensing noise is absent ($\sigma^{sen} = 0$), and a maximum likelihood decoder is applied to a 1D multiperiodic code, the Fisher information increases with narrower tuning widths, Figure 2.4A. This increase, as for the single-population codes, can be directly attributed to the increase in the coding line length with decreasing tuning width. With zero sensing noise, the probability of threshold error (numerically estimated) for the multiperiodic code remains extremely close to zero, Figure 2.4B, regardless of tuning curve width in the regime we consider. Thus, in terms of both Mutual information and mean-squared error, the error is fully specified by the FI, and the narrower the tuning curves, the better, Figure 2.4D and E.

This result on multiperiodic multi-population codes is qualitatively the same as the findings for unimodal bump codes both for unbounded variables [105] and circular variables [17, 65] in one dimension.

When the sensing noise σ^{sen} is non-zero (independent additive Gaussian noise per phase, followed by the modulo operation to keep the phase variable within [0,1)), the increase in FI with narrower tuning saturates when the tuning curves become narrower than σ^{sen}, Figure 2.4A. Once again, this result is similar to the single-population case. In contrast to the single-population case, however, is the threshold error probability. For a fixed tuning curve width, as the sensing noise is varied, the probability of threshold errors abruptly jumps to

a large non-zero value for sufficiently large sensing noise, Figure 2.4B (around $\sigma^{sen} = 0.2$). For a fixed value of sensing noise, there is a weaker variation in the threshold error probability with tuning curve width: threshold errors become more probable as the tuning curve width *increases*. This result cannot be attributed to any property of single-population coding; it is inherently due to the multiperiodic nature of the multipopulation code.

The weak growth in the probability of threshold errors with tuning curve width augments the effects of decline of FI with wider tuning, to result in a slightly stronger overall decline in MI with wider tuning than in the single-population case.

We may understand these effects through a geometric view of coding. To do so, first note that the maximum likelihood estimate of x given the spike counts K of all neurons equals the maximum likelihood estimate of x from the maximum likelihood estimate $\hat{\vec{\phi}}$ of the phases in all populations. This is because within each population, $\hat{\vec{\phi}}$ is a sufficient statistic for x given K. Given this fact, the estimation of x becomes a two-step process, only one of which depends on tuning curve widths: estimation of each phase, $\hat{\phi}_n$, given the spikes K_n from that population, followed by estimation of x given the phases. The first problem is simply a single-bump code decoding problem for a circular variable (spatial phase within a population), and the standard results on FI as a function of tuning curve width apply [17, 65].

The second type of errors are *threshold errors* [81]: a sufficiently large perturbation can cause the decoder to map the perturbed codeword to the

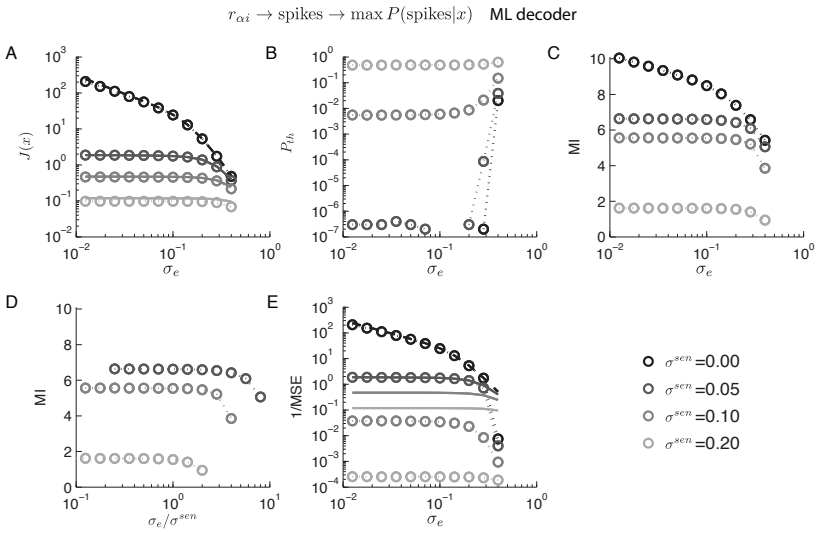

Figure 2.4: Ideal (ML) decoding and dependence on tuning curve width ($D = 1$). In the presence of sensing noise, tuning curve width (σ_e) has little effect on the performance of the ML decoder. (A) If $\sigma^{sen} > 0$, Fisher information about the spatial location increases marginally as σ_e decreases (solid curves). This contrasts the large increase in Fisher information for a narrower σ_e when $\sigma^{sen} = 0$ (dashed curve). FI is numerically calculated (circles) by the inverse of variance of the center peak, which agrees with analytical predictions (curves). The size of the sensing error determines the maximum FI that can be achievable. (B) The probability of threshold error stays relatively constant and marginally increases for too wide tuning curve width. When the sensing noise is too large ($\sigma^{sen} = 0.20$), the probability of threshold error becomes large (top circles). Otherwise, the probability of threshold error stays close to zero. (C) The mutual information (MI), calculated by numerically calculating posterior probabilities, shows the same pattern as FI in (A). (D) MI is shown for $\sigma^{sen} > 0$ with the x-axis normalized by σ^{sen}. When $\sigma_e < \sigma^{sen}$, MI is flat. When $\sigma_e > \sigma^{sen}$, MI slightly decreases. (E) A larger σ^{sen} results in a larger mean square error (MSE). For a fixed σ^{sen}, MSE increases with a wider σ_e. Circles represent numerically calculated MSE while lines indicate the inverse of the analytical FI. The former deviates from the latter when σ^{sen} is large due to threshold errors. Parameters used for numerical simulations are as follows: $R_l = 300$, $H = 19200$, $N = 8, \lambda_n = \{30, 34, 38, 42, 46, 50, 54, 58\}$, $M = 256$, and $r_m \Delta t = 1$. σ_e and σ^{sen} are independently varied within ranges $[0.0125, 0.2828]$ and $[0, 0.2]$, respectively. For each condition, 10^7 samples are drawn independently.

wrong segment of the embedding, resulting in a much larger error, out of proportion to the size of the perturbation, Figure 2.2. These errors occur occasionally.

Both local and threshold errors contribute to a finite mutual information between the coded variable and its decoded estimate. The width of neural tuning, because it affects the embedding of the coding line into neural activity space, shapes the effects of noise on decoding precision.

First, we focus on 1-dimensional variables. As expected from previous studies of population coding [10, 69, 77, 105] with unimodal tuning curves, the Fisher information contained about network phase ϕ_n in a finite sample of spikes from one network grows as the tuning curve width shrinks, Figure 2.4A. In such codes, the embedding of the coding line in the M-dimensional neural activity space is so sparse that the probability of threshold errors is nil, and all errors are local errors that, for large numbers M of neurons, are Gaussian. This error is then well-quantified by the inverse Fisher information through the Cramer-Rao bound [69, 77]. Analytical calculations of the Fisher information for single-population codes show that for 1-dimensional variables, the information in a homogeneous neural population whose tuning curves are identical up to simple shifts, grows with decreasing tuning curve width [10, 105]. Previous results, as well as the dashed curve in Figure 2.4A, are based on the existence of spike sample errors, but in general do not include sensing errors. The solid curves in Figure 2.4A incorporate the existence of increasing amounts of sensing errors, and show the dependence of Fisher informationMutua in each of the

networks about the network phase (See Appendix A.3 for detail calculation). The probability of threshold errors with a maximum likelihood decoder is zero when the sensory noise is small, but increases sharply for increasing amounts of sensory noise (lighter gray curve with $\sigma^{sen} = 0.20$) beyond a threshold, Figure 2.4B. Threshold error probability is only weakly dependent on tuning curve width, with a slight decrease for wider tuning curves.

In Figure 2.4C, we may see the overall effect of tuning curve width on mutual information, which is estimated numerically by applying maximum likelihood decoding to samples of the noisy codewords. Without sensing errors, there are continuing gains in information progressively narrower tuning curves (assuming there are still enough neurons to cover the entire coding space). With sensing errors, there is no real advantage to making the tuning curves much narrower than the standard deviation of the sensing error. However, up to that point, narrower tuning curves are better.

2.3.5 Finite tuning curve widths σ_e minimize errors with neural network decoding with narrow σ_h

Next, we consider the effects of tuning curve width on the ability of biologically plausible neural network decoders to estimate the value of the encoded variable. The neural network decoder consists of a single-layer feed-forward stage, followed by a max operation, to identity the maximally driven output unit. This unit is then active, while the rest are inactive. Each output unit represents a possible value of the coded variable, through its binary

activation (if one output unit is active, then the inferred value of the coded variable is the preferred value for that unit), and only one output unit can be active at a time, selected by the max operation.

The weights from the inputs to each output unit reflect the *template* input for the output's preferred location: they are learned by clamping the inputs in a state reflecting the accurate encoding of the external variable (no sensing or spike sampling errors), and the appropriate output unit is clamped on. The weights are set by simple associative learning, to be the product of the activations of the pre- and post-synaptic units. The procedure is repeated so all output units are clamped on once, and their weights trained (See Appendix A.1).

How does the estimation error of the NN decoder scale with tuning curve width, and how does the overall estimation error compare with the optimal (ML) decoder? The neural network decoder displays a significantly different behavior in the threshold error probability for narrow tuning curves. Whereas the threshold error probability continues to modestly decline with decreasing tuning curve width for the ML decoder, it decreases then sharply increases for narrower tuning with the NN decoder, Figure 2.5B. That is, for very narrow tuning curves, the probability of threshold error grows steeply (for any curve that includes sensing errors). As a result, the mutual information between the multi-period population responses and the encoded variable grows smoothly then declines sharply as the tuning curves are narrowed, whenever there is non-zero sensing error (gray squares), Figure 2.5C.

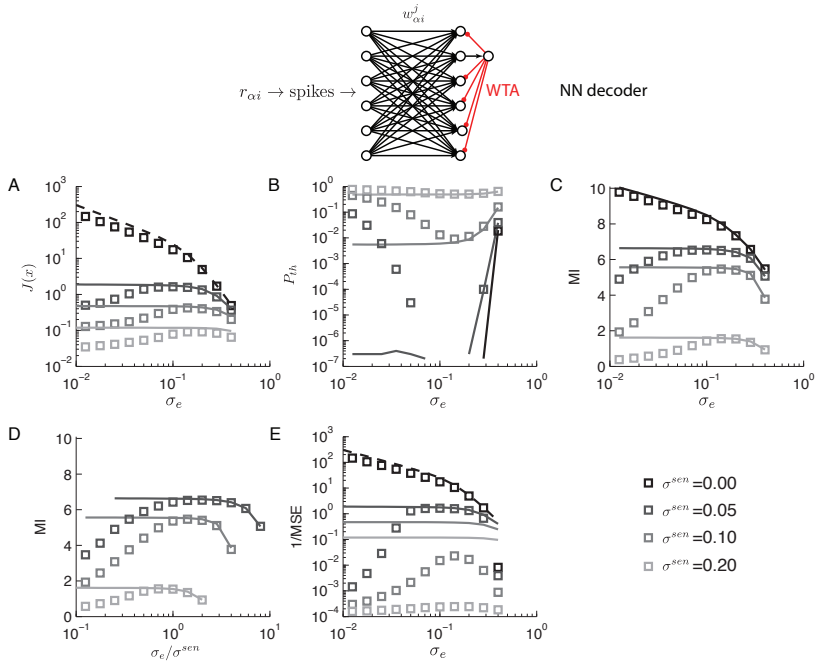

Figure 2.5: Neural network decoding and dependence on tuning curve width ($D = 1$). The NN decoder trained with a narrower tuning curve at the decoder ($\sigma_h \sim \frac{1}{H} \ll 1$) performs poorly when tuning curve width is narrow. The results by the NN decoder is shown by squares. Solid or dashed curves show analytical predictions (A,E) or results for ML in Figure 2.4 (B-D) for comparison. (A) When $\sigma^{sen} = 0$, numerically calculated FI (squares) is close to analytical prediction (dashed curve). However, when $\sigma^{sen} > 0$, numerically calculated FI (squares) drops from the analytical prediction (solid curve) for narrower tuning curve widths. This is in contrast to marginally increasing FI as σ_e decreases with ML decoder. (B) In contrast to the ML decoder, the NN decoder produces a large threshold error probability when turning curve width is narrow in the presence of sensing error. (C) Consequently, when $\sigma^{sen} > 0$, the total mutual information (MI) by the NN decoder (circles) drops precipitously for narrow tuning curve widths, deviating from the ML decoder (solid curves). (D) MI for $\sigma^{sen} > 0$ is replotted after normalizing the x-axis by σ^{sen}. When $\sigma_e < \sigma^{sen}$, MI rapidly drops. When $\sigma_e > \sigma^{sen}$, MI slightly decreases. (E) For a fixed $\sigma^{sen} > 0$, MSE rapidly increases for a narrower σ_e and gradually increases for a wider σ_e. Parameters used for numerical simulations are as follows: $R_l = 300$, $H = 19200$, $N = 8, \lambda_n = \{30, 34, 38, 42, 46, 50, 54, 58\}$, $M = 256$, and $r_m \Delta t = 1$. For each condition, 10^5 samples are drawn independently.

2.3.6 Learning with noisy data improves NN decoding

Next, we modified the neural network decoder to incorporate the sensing noise during weight training. The sensing noise in GC responses results in smoothing in the trained weight w_{ni}^{j}. This is equivalent to increasing the decoding tuning curve width σ_h (See Appendix A.7). Thus, in numerical simulations, we increased σ_h and used analytical weight capturing the smoothing effect.

We find that overall, the total information between decoded estimate and the external variable increases when the feedforward (decoding) weights are learned in this way, Figure 2.6C, relative to when they are learned with no sensing noise, Figure 2.5C. This is mainly the result of decreased threshold error probability. In Figure 2.6B, note that large threshold error when $\sigma_e \ll \sigma_{sen}$ is rescued by increasing σ_h. However, as σ_h increases marginal gain decreases.

Indeed, the major gains in information occurred for narrow tuning curves: performance with the NN decoder with weights learned in the presence of sensing noise approaches the performance of the ML decoder from below, showing that this NN decoder approximates ML decoding. In particular, the penalty incurred for narrow tuning curves by a NN decoder trained without sensing noise, is largely removed, so that beyond a certain sparseness, the NN decoder performance saturates, rather than decreasing, but it also fails to increase as tuning curves are further narrowed, Figure 2.6D and E.

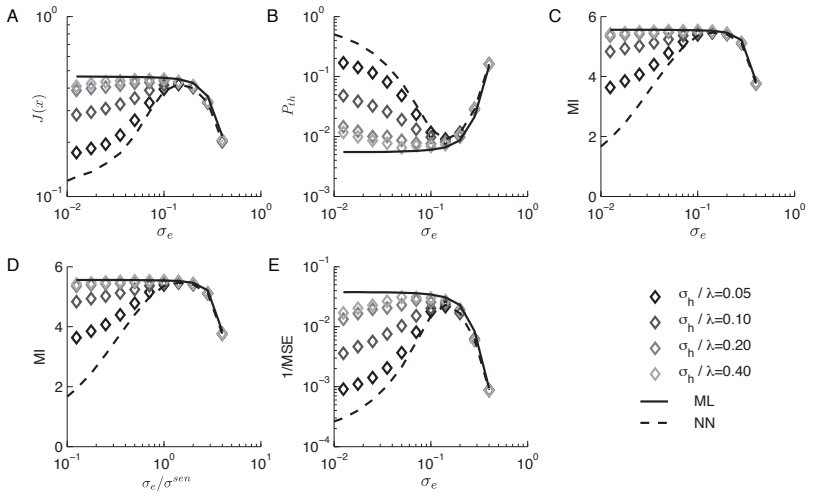

Figure 2.6: The catastrophic decoding failure of the NN decoder is alleviated by increasing the tuning curve of the decoder σ_h ($D = 1, \sigma^{sen} = 0.1$). (A) As σ_h increases, FI increases close to analytical prediction (solid curve). (B) As σ_h increases, threshold error for a narrow σ_e decreases from the level of the NN decoder (dashed curve at the top) to that of ML decoder (solid curve at the bottom). (C) Consequently, with increased σ_h, the total mutual information (MI) decreases slowly for a narrow tuning curve width (solid curve). (D) MI is replotted after normalizing the x-axis by σ^{sen}. The NN decoder with a large σ_h provides similar performance as the ML decoder. (E) MSE becomes close to the MSE of ML (solid curve) compared to that of NN (dashed curve). Still, there is a gap for a very narrow tuning curve. Parameters used for numerical simulations are as follows: $R_l = 300$, $H = 19200$, $N = 8, \lambda_n = \{30, 34, 38, 42, 46, 50, 54, 58\}$, $\underline{\lambda} = 44$, $M = 256$, and $r_m \Delta t = 1$. The number of samples for each condition is 10^5.

2.3.7 Geometric underpinnings of results

Why does the neural network decoder fail for narrow tuning curves, by giving large threshold errors? First, let us turn to a concrete picture of the multi-scale code and its decoding, in the space of the coded variable, Figure

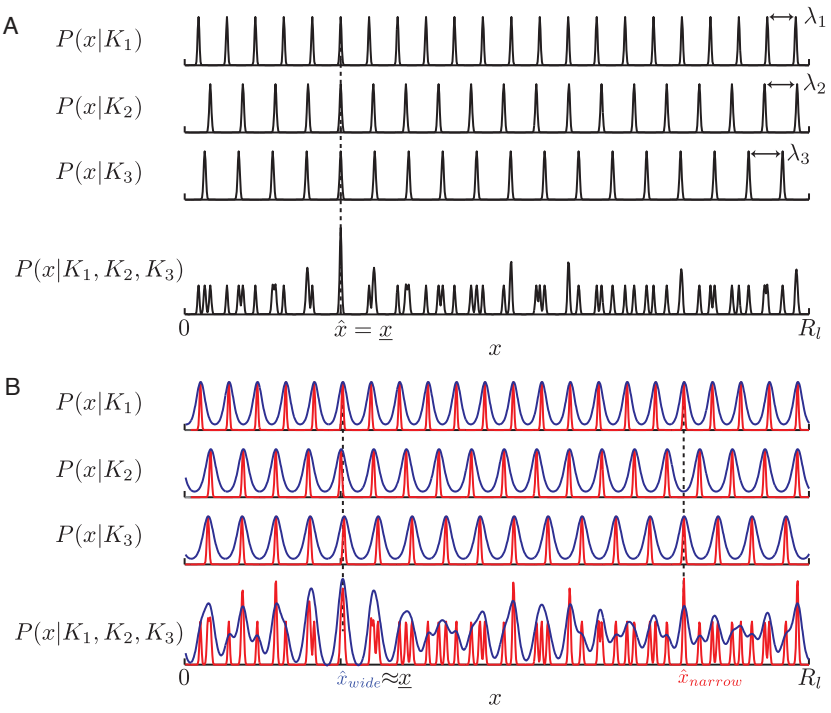

Figure 2.7: Why NN decoding fails for narrow (encoder) tuning widths. (A) The first three rows represent the probability of the spatial location given neural responses of each network K_n without the sensing error. In individual networks, any location that has the same phase as the true location ($\underline{x}+m\lambda_n$, m is an integer) is equally likely to produce the same neural response and cannot be distinguished solely from the neural responses from a single network. This ambiguity is resolved by observing all the grid networks. In the bottom, the posterior has a unique maximum at the true location (gray dashed line in the left). (B) In contrast, in the presence of the sensing error, the posteriors in each network is shifted independently (red in the first three rows). The combined posterior (red in the bottom) produces maximum in a complete different location, which produces an estimate $\hat{x}_{narrow}$. This sensitivity to sensing error is reduced by increasing the width of the posterior (blue). The same noise that causes the complete failure of estimation for a narrow tuning curve (dark) results in the estimate $\hat{x}_{wide}$ with a negligible error for a broader posterior (blue in the bottom).

2.7. When there is no sensing noise, then the activity peaks of all the different networks align at a specific value of the coded variable x, which corresponds to the true value of the external variable, Figure 2.7A. It is easy to see that the width of tuning then determines the local precision of the estimate: the narrower the tuning in each network, the tighter the cumulative set of aligned peaks, and the higher the precision of the estimate (as in Figure 2.5C above). However, when sensing noise is introduced, the peaks of each population shift relative to the true location, and relative to the peaks of the others. If the peaks are narrow (narrow or red curves, Figure 2.7B), even small shifts cause a total misalignment at the true location. Peaks of a subset of different networks may come into close alignment, but typically do so at locations far from the true location, Figure 2.7B. A decoder that simply computes the maximum of the peaks summed across networks (as does a winner-take-all neural network) will decode location as being at one of these remote locations of close alignment of a fraction of networks. This corresponds to a threshold error, and is responsible for the steep increase in error probability for neural network decoding when sensing errors are introduced.

With broader tuning (broad or blue curves, Figure 2.7B), small shifts in phase do not cause a total misalignment of peaks at the true location. On the contrary, the peaks from all N networks continue to have some overlap at the true location, and when summed, produce maximal activation in the immediate neighborhood of the correct location, up to the uncertainty inherent in the width of the cumulative activity peak. At the locations where a subset

of networks have very well-aligned peaks, the summed activation is now lower than at the true location, because some networks are entirely lacking a peak there, and cannot contribute to the overall sum. Thus, NN estimates applied to tuning widths that approach the size of sensing errors are less likely to have threshold errors. However, broadening the tuning curves much beyond the expected shifts in phase across networks will produce little further benefit in reducing threshold errors, but will cause a decline in the precision of the estimate of location in the neighborhood of the true location.

A maximum likelihood decoder does not simply compute the maximum of the peaks, summed across networks. ML decoding incorporates the fact that there is sensing noise with standard deviation σ^{sen}. The ML decoder effectively convolves the peaks from individual networks with a Gaussian of width σ^{sen}, before summing them together in (A.5). Therefore, a ML decoder takes into account the shifts induced by sensing noise, effectively broadening the tuning curves by that amount, in the decoding process. This effective broadening exactly accounts for the effects of sensing noise, and minimizes the probability of threshold errors in decoding. Thus, for an ML decoder there is no reduction in threshold error from narrowing the tuning curves themselves; and, as always, broadening contributes to a decline in estimation precision in the neighborhood of the true location. Thus, with ML decoding, narrower tuning is better.

Next, let us examine a different representation of the coding problem, by considering the embedding of the multi-period population code into the

high-dimensional activity space of neurons. The solid curves in Figure 2.8A-C illustrates noise-free firing rates of three neurons for different tuning curve width, defining the embedding of a low-dimensional input variable to a higher dimensional state space. When tuning curve width is narrow (left), neurons are active for a small range of inputs and, therefore, the embedding curves are close to axes. As tuning curve width becomes wider (middle), the embedding curves becomes better separated. However, a too wider tuning results in a poor packing (right) because all the neurons are active for any input variable.

Next, let us compare estimated states by different decoders. Black cross and red dots in Figure 2.8A-C shows true and noisy neural responses, respectively. Circles in A,B, and C represent states estimated by ML, NN with narrow σ_h, and NN with wide σ_h, respectively. Dashed lines connect a noisy state to its estimate. In Figure 2.8A, with ML decoder, estimated states are close to the true state *along* the embedding curve. However, in Figure 2.8B, NN with narrow σ_h decoder maps noisy states to a closest point on the embedding curve, which fails to capture the actual noisy characteristics and poor estimation with narrow tuning (left). Increasing σ_h overcomes this failure and estimated states become closer to those of ML, Figure 2.8C.

Finally, to further understand the dependence on decoding method, let us look at the state space of phase. Figure 2.9 illustrates the multi-scale code with $\lambda = \{31, 33\}, R_l = 495, M = 1024, \sigma^{sen} = 0.05, \sigma_e = 0.001,\ r_m dt = 50$. Here, M, the number of neurons per a network, is chosen to be large to reduce the effect of the Poisson variability compared to the sensing error. Solid curves

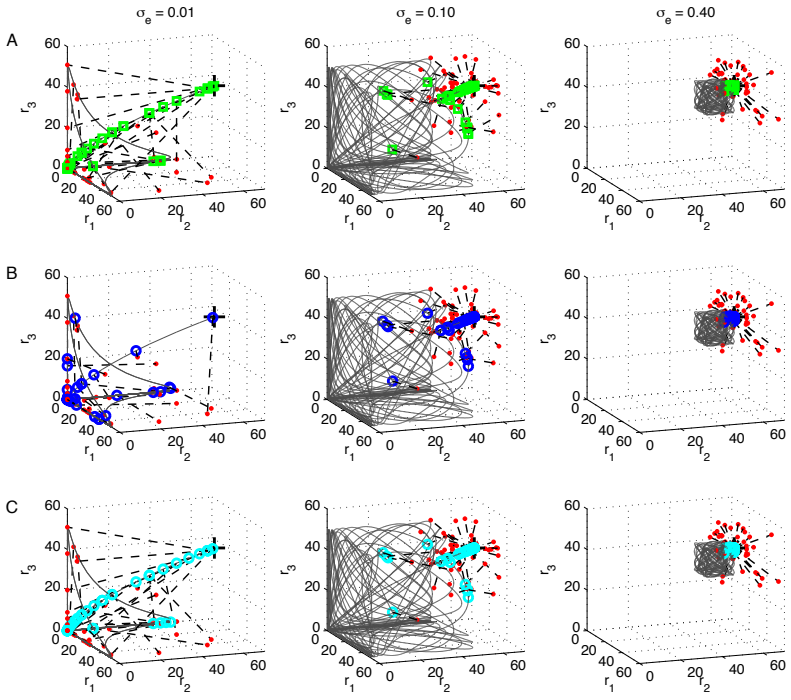

Figure 2.8: The NN decoder becomes closer to the ML decoder by increasing tuning curve width of the decoder (σ_h) for a narrower GC tuning curve (σ_e). Solid curves represent noise-free neural representations of spatial location $x \in [-500, 500]$ with $\lambda = \{31, 33, 37\}, M = 256, r_m \Delta t = 50$ with black crosses corresponding to true spatial location $x = 0$. The number of neurons in the NN decoder (H) is 20000. Red dots show 50 samples of noisy grid cell spike counts due to sensing ($\sigma^{sen} = 0.05$) and spike sample (Poisson) noises. Green, blue, cyan circles mark neural representations corresponding to decoded location by the ML decoder (A), the NN decoder with narrow $\sigma_h \sim 1/H \ll \sigma_e \underline{\lambda}$ (B), and the NN decoder with a wider $\sigma_h = 0.1\underline{\lambda}$ (C), respectively. Dashed line connects each noisy response to its estimate. The tuning curve widths are $\sigma_e = 0.01, 0.1, 0.4$ from the left to the right. With the ML decoder, most of estimates are close to the true value (A) regardless of σ_e. In contrast, the NN decoder produces wrong estimates when σ_e and σ_h are both small (B left). This catastrophic decoding failure is rescued by increasing σ_h (C left).

show the embedding curve of the input range to the phase space. Black cross and red dots represent true and noisy phases due to both sensing and spike sample errors. The same noisy responses are decoded by ML (A), NN with narrow σ_h (B), and NN with wide σ_h (C). Compared to the ML decoder (A), the NN decoder with narrow σ_h (B) occasionally produces large errors in estimating one of the phases, consistent with the higher threshold probability of the NN decoder with narrow σ_e and σ_h. Increasing σ_h, corresponding to noisy learning, fixes this problem (C).

2.4 Conclusion

Motivated by the unusual periodic responses of grid cells to animal location, and the existence of multiple spatial periods within a single individual, we analyze the mutual information between an encoded variable (location) and a set of neural populations with periodic responses of diverse periodicity. Beyond independent Poisson spiking noise, we also consider the role of latent variables that contribute systemic, correlated errors within each population.

We show that with an ideal (maximum likelihood) decoder, the tuning curves should be as narrow as possible to maximize mutual information, similar to the result for homogeneous, single-bump population codes for periodic variables. However, for plausible neural-network decoding, the optimal tuning curve width is finite and independent of the dimension of the coded variable. This result is in contrast to findings for single-bump population codes, and is due to the existence of non-local errors in the posterior probability distribu-

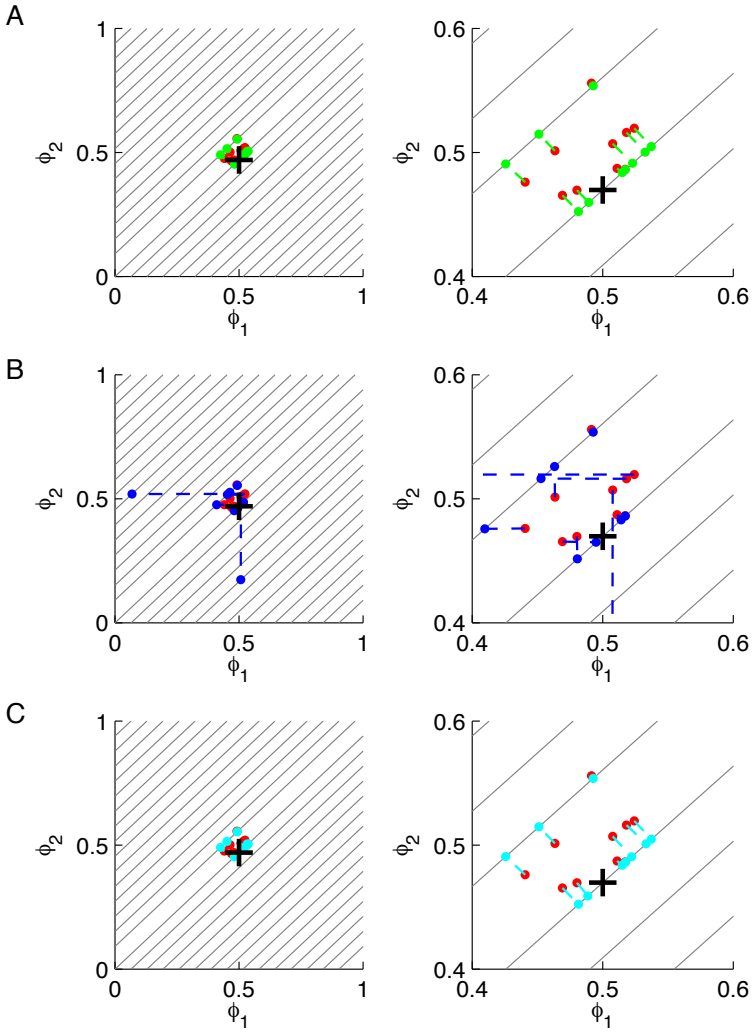

Figure 2.9: Decoding by ML (A), NN with narrow σ_h (B), and NN with wide σ_h in the state space of phase. The right is the magnified plot of the left. Gray lines represent noise-free phase, the codebook. Black cross and red dots represent true and noisy phases, respectively. Green, blue, and cyan represent estimated phases by different decoders. Dashed lines connect noisy and estimated phases.

tion of estimated location given the noisy neural code. We show that broad tuning curves in the neural network decoder can significantly mitigate the loss in performance from narrow tuning in the encoding population, and show how appropriately broad decoder tuning can be adaptively acquired through associative learning.

Chapter 3

Decoding multi-scale error-correcting codes using belief propagation

3.1 Introduction

This chapter focuses on recovering self-location from noisy phases of grid cell modules. Grid cells in the entorhinal cortex encode self-location as a set of phases using distinct moduli. The codewords of such multi-scale codes range over the N-dimensional phase domain with complex structure. Thus, decoding multi-scale codes is a challenging problem. To solve this, we introduce an inference algorithm studied for graphical models.

Estimating sources from a given set of noisy measurements of its residues is a mathematical formulation of a problem originating in multiple disciplines. For example, in coding theory, redundant residual number system (RRNS) codes [84] are designed to encode an integer into residues modulo relatively prime integers and find applications in storage and related domains. Our work builds on the effort towards efficient decoding algorithms for such codes [8, 54, 102]. Further applications for a similar problem setting are found when performing decentralized compression in sensor networks [34], as well as in mechanisms for distributed consensus in social networks [68].

Whereas the integer version of the original RRNS code is easily decoded by a closed form expression with appropriately designed moduli [8,54,102], estimating continuous source from continuous residues is a much more challenging problem, one that requires sophisticated tools such as belief propagation for efficient (approximate) solutions.

In this chapter, we first demonstrate that this noisy source estimation problem is indeed equivalent to a suitable integer program, and subsequently develop an iterative inference algorithm over an appropriately defined graph. Our approach is motivated by the clever use of graphical model framework to solve systems of linear equations as studied in [82]. For a given system of linear equations, a corresponding graphical model is first constructed and then the Gaussian belief propagation algorithm is performed over the graph to find the solution. Consequently, [82] shows that a large-scale linear optimization can be solved by iteratively exchanging local messages between neighboring nodes in a bipartite graph. This framework is extended to estimating noisy sources from simultaneous continuous-valued congruences over a suitably defined layered graph, and shows that such an algorithm is convergent and effective at high SNRs.

Since its introduction by Pearl [71], belief propagation (BP) has enjoyed great successes in efficiently calculating maximum a posteriori (MAP) estimate of hidden variables given noisy or partial measurements. One of the most important applications of BP is decoding modern channel codes, such as turbo codes [12, 63] and low density parity check codes [27, 40]. (Refer to [75] and

references therein for more details.) Theoretical studies on how and when BP works are active research areas. If the graph over which BP is performed is cycle-free, convergence and correctness of BP are guaranteed [71]. BP is also found, on multiple occasions, to perform well on complex graphs with cycles. Weiss and Freeman study BP on graphs with arbitrary topologies and show that the fixed-point of the max-sum algorithm is the same as the MAP estimate over a wide range of neighboring configurations (which is stronger than local optimality but weaker than global optimality) [96]. Bayati *et al.* apply the max-sum algorithm to maximum weigh matching in a bipartite graph and show the convergence and correctness [9]. In our work, we present a belief propagation algorithm that can be understood as an extension of the framework in [9] in the sense that our algorithm operates on a graph with more than two partitions.

3.2 Problem definition

A source $S \in [0, 1)$ is encoded to an N-dimensional vector $\mathbf{X}$ which contains residues of S with respect to distinct moduli:

$$\mathbf{X}(S) = (X_1(S), X_2(S), \ldots, X_N(S)) \tag{3.1}$$

$$X_n(S) = a_n S \mod 1, \quad n = 1, 2, \ldots N, \tag{3.2}$$

where a_n's are relatively prime integers, $\gcd(a_n, a_m) = 1$ if $n \neq m$. The goal of the decoder is to estimate the source S from the noisy measurement

$\mathbf{Y} = (Y_1, Y_2, \ldots, Y_N)$ of the residue $\mathbf{X}$:

$$Y_n = X_n + Z_n \tag{3.3}$$

$$Z_n \sim \mathcal{N}(0, \sigma^2), \tag{3.4}$$

where Z_n's are additive Gaussian noise with zero mean and variance σ^2, independent of one another. In other words, we would like to estimate the source S given N simultaneous congruences with additive noise.

3.3 The maximum likelihood estimation of the source from noisy residues

3.3.1 The maximum likelihood (ML) estimation involves maximization of a non-convex function.

When the source is equally likely in $[0, 1)$, the maximum a posteriori (MAP) estimate coincides with the maximum likelihood (ML) estimate:

$$\hat{S}_{ML} = \arg\max_{S\in[0,1)} P(\mathbf{Y}|S) = \arg\max_{S\in[0,1)} \prod_{n=1}^{N} P(Y_n|S), \tag{3.5}$$

where the product form in (3.5) follows from the independence of noise.

Each likelihood term $P(Y_n|S)$, $n = 1, 2, \ldots N$, and total likelihood $P(\mathbf{Y}|S)$ in (3.5) are expressed as mixtures of Gaussians (MoG) as follows. Due to the modulo operation, the likelihood from the n'th congruence $P(Y_n|S)$ is a periodic function and each period is the Gaussian with a different mean and the identical variance (Figure 3.1A). Thus, $P(Y_n|S)$ is a mixture of Gaussians:

$$P(Y_n|S) = \sum_{m=0}^{a_n-1} \mathcal{N}\left(\frac{y_n}{a_n} + \frac{m}{a_n} \bmod 1, \frac{\sigma^2}{a_n^2}\right), \tag{3.6}$$

where m indexes the period and $\mathcal{N}(\mu, \nu)$ denotes the probability distribution function of Gaussian with mean μ and variance v:

$$\mathcal{N}(\mu, \nu) = \frac{1}{\sqrt{2\pi\nu}} e^{-\frac{(S-\mu)^2}{2\nu}}. \tag{3.7}$$

Thus, the total likelihood $P(\mathbf{Y}|S)$ is the product of mixtures of Gaussians (Figure 3.1B).

$$P(\mathbf{Y}|S) = \prod_{n=1}^{N} \left(\sum_{m=0}^{a_n - 1} \mathcal{N}\left(\frac{y_n}{a_n} + \frac{m}{a_n} \bmod 1, \frac{\sigma^2}{a_n^2} \right) \right). \tag{3.8}$$

Therefore, the maximum likelihood estimation in (3.5) is equivalent to the maximization of the non-convex objective function in (3.8):

$$\hat{S}_{ML} = \arg\max_{S} \prod_{n=1}^{N} \left(\sum_{m=0}^{a_n - 1} \mathcal{N}\left(\frac{y_n}{a_n} + \frac{m}{a_n} \bmod 1, \frac{\sigma^2}{a_n^2} \right) \right). \tag{3.9}$$

Note that each $P(Y_n|S)$ produces a_n equally likely maxima (Figure 3.1A) and, consequently, $P(\mathbf{Y}|S)$ has $\prod_{n=1}^{N} a_n$ local minima (Figure 3.1B).

3.3.2 The maximum likelihood (ML) estimation is a combinatorial problem.

A naïve way to implement the ML estimator in (3.9) would be to calculate the likelihood of densely quantized bins in the source domain and then to search for the bin with the maximum value. The accuracy and the speed of this approach would be limited by the bin size. The denser the bins are, the more accurate the estimate is. However, this performance increase comes at a price of high computational complexity and large memory size.

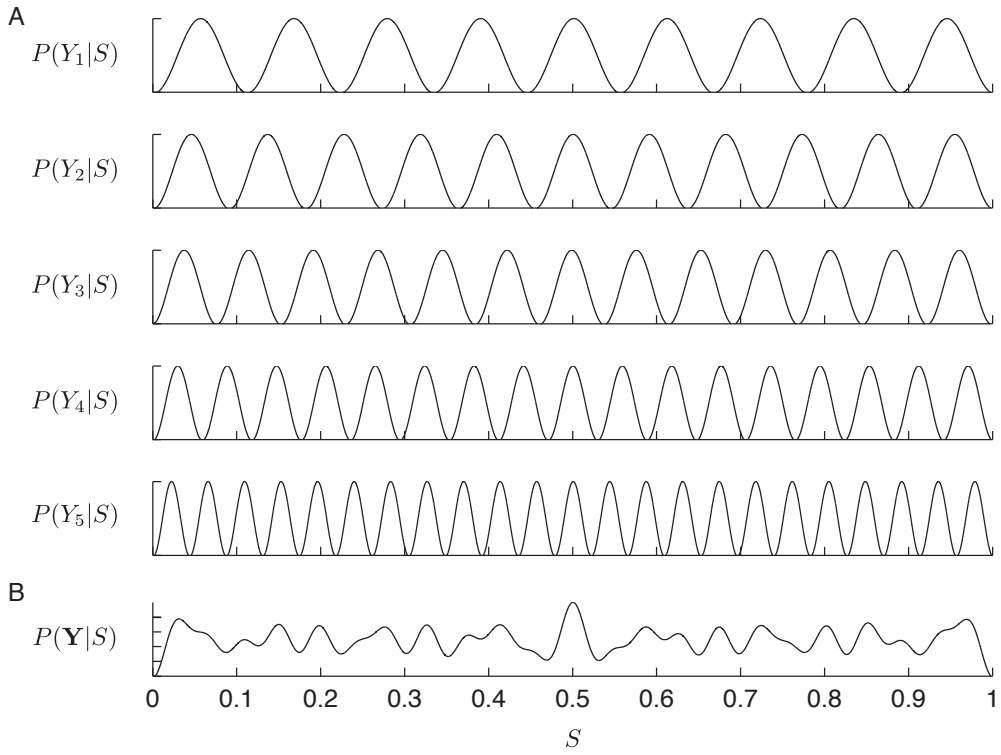

Figure 3.1: Likelihoods from individual measurements (A) and combined likelihood (B) for $S = 0.5$ and $a_n = \{9, 11, 13, 17, 23\}$. In A, each measurement Y_n $(n = 1, 2, \ldots, N)$ produces a periodic likelihood $P(Y_n|S)$ in (3.6) which is a mixture of Gaussian. In B, the combined likelihood $P(\mathbf{Y}|S)$ in (3.8), which is the product of the individual likelihoods in A, is a non-convex function with $\prod_{n=1}^{N} a_n$ local maxima.

Instead, the quantization of the entire source range can be avoided by keeping track of mean and variances of MoGs. Each local maximum of the total likelihood in (3.8) results from local maxima of individual likelihoods (3.6). Let m_n index the local maxima in $P(Y_n|S)$. Then, the local maxiam in $P(\mathbf{Y}|S)$ is identified by $(m_1, m_2, \ldots, m_N)$, which forms the set of integer

points:

$$\mathcal{M} = \{(m_1, m_2, \ldots, m_N) \mid 0 \leq m_n < a_n,\ m_n \in \mathcal{Z}\}\,. \tag{3.10}$$

The likelihood corresponding to $\mathbf{m} \in \mathcal{M}$ is calculated by combining the m_n' th Gaussian from congruence n. This calculation is done by recursively combining two Gaussians at a time (See Appendix B.1 for combining a pair of Gaussians).

Therefore, the ML estimation is equivalent to the following integer programming:

$$\mathbf{m}^* = \arg \max_{(m_1,m_2,\ldots,m_N)\in\mathcal{M}} \prod_{n=1}^{N} \mathcal{N}\left(\frac{y_n}{a_n} + \frac{m_n}{a_n} \pmod 1, \frac{\sigma^2}{a_n^2}\right), \tag{3.11}$$

where $\mathbf{m}^* = (m_1^*, m_2^*, \ldots, m_N^*)$ denotes the optimal combination of MoGs that produces the maximum likelihood. After identifying the optimal index set $\mathbf{m}^*$, the estimate of the source is the mean of the product of Gaussians chosen by $\mathbf{m}^*$:

$$\hat{S}_{MoG} = \left(\sum_{n=1}^{N} \frac{\mu_n^*}{\nu_n}\right)\left(\sum_{n=1}^{N} \frac{1}{\nu_n}\right) \tag{3.12}$$

$$\mu_n^* = \frac{y_n}{a_n} + \frac{m_n^*}{a_n} \pmod 1 \tag{3.13}$$

$$\nu_n = \frac{\sigma^2}{a_n^2}, \tag{3.14}$$

where μ_n^* and ν_n are mean and variance from $\mathbf{m}^*$ found in (3.11).

The second approach of keeping only mean and variance is more efficient than the naïve method based on dense quantization of the source domain. Still, the optimization (3.11) involves a search over an exponentially increasing range

(with respect to N) and the objective function is not convex. In what follows, we approximate (3.11) with a simpler form and present a belief propagation algorithm that makes use of local computations to efficiently estimate the source.

3.4 Resolving simultaneous congruences by belief propagation

3.4.1 From global to local computations

First, the integer programming in (3.11) is simplified by reducing the number of congruences considered at a given time. In (3.11), the search domain $\mathcal{M}$ is a N-dimensional space and, therefore, the direct evaluation of the likelihood for a given $\mathbf{m} \in \mathcal{M}$ involves all the N congruences. Instead, we derive local computations involving only a pair of congruences at a given time.

Specifically, let us introduce the following graphical model as shown in Figure 3.2A. The a_n possible configurations of m_n is represented by a_n indicator nodes; the m_n'th node is 1 while the remaining nodes are 0. These a_n nodes, corresponding congruence n, is called *layer* n and denoted as L_n. Each row in Figure 3.2A is a layer and there are total N layers and $\sum_{n=1}^{N} a_n$ nodes. For notational simplicity, we use *single* index $i = 1, 2, \ldots, \sum_{n=1}^{N} a_n$ for nodes. Nodes i and j in two different layers are connected by an edge with weight s_{ij}. Nodes within the same layer are not allowed to be connected. Potentially, edges could be defined for $\binom{n}{2}$ pairs of layers. However, this would increase computational complexity. Instead, we allow edges between consecutive layers

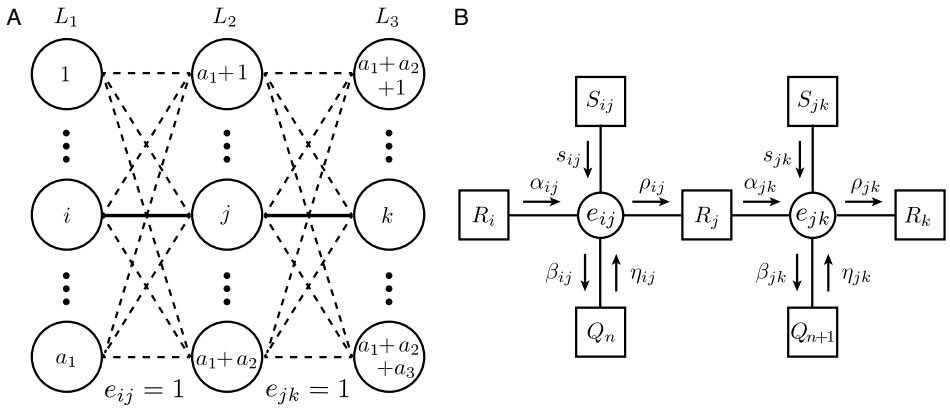

Figure 3.2: (A) Graphical model representation of three simultaneous congruences. Layer L_n corresponds to nth congruence relationship. Binary hidden variable e_{ij} indicates whether a pair of Gaussians i and j are selected or not. (B) Factor graph representation is shown for two variable nodes e_{ij} and e_{jk} with corresponding messages. Constraint Q_n is imposed on all the edges from layer $n-1$ to n and constraint R_j check the consistency on node j.

only. The weight s_{ij} between node i and j reflects the similarity between the two nodes, defined as follows:

$$s_{ij} = \log \kappa_{ij} \quad = \quad \frac{1}{2}\left\{ \frac{\mu_i^2}{\nu_i}\left(\frac{\nu_{ij}}{\nu_i}-1\right) + \frac{\mu_j^2}{\nu_j}\left(\frac{\nu_{ij}}{\nu_j}-1\right) + 2\frac{\nu_{ij}}{\nu_i\nu_j}\mu_i\mu_j + \log\left(\frac{\nu_i\nu_j}{\nu_{ij}}\right)\right\}, \tag{3.15}$$

where κ_{ij} is the magnitude of the the product of two Gaussians corresponding node i and j, $\mathcal{N}(\mu_i, \nu_i)$ and $\mathcal{N}(\mu_j, \nu_j)$ (See (B.4) in Appendix B.1).

Instead of considering nodes across *all* layers simultaneously, the optimal set of nodes is inferred by locally evaluating for a *pair* of layers based on the similarities defined in (3.15). For this purpose, we introduce a binary variable $e_{ij} \in \{1, 0\}$ to indicate whether *both* node i and j are selected ($e_{ij} = 1$) or not ($e_{ij} = 1$). Let $\mathcal{E}$ denote the set of all possible configurations of edges. Then, the goal is to find the optimal configuration $E^* \in \mathcal{E}$ that maximizes the sum

weight of chosen edges, namely $\sum_i \sum_j s_{ij} e_{ij}$. The parameterization of a congruence by multiple binary variables expands the search space but introduces constraints on the problem. Note that the size of the possible configurations $\mathcal{E}$ is $\prod_{n=1}^{N} 2^{a_n}$, which is larger than the ways of choosing one node per each layer, $\prod_{n=1}^{N} a_n$. This relaxation, which appears to be more costly, actually simplifies local messages in belief propagation and offers better interpretations[1].

Specifically, we have the following constraints:

$$Q_n = \begin{cases} 0 & \text{if } \sum_{i \in L_{n-1}, j \in L_n} e_{ij} = 1 \\ -\infty & \text{otherwise,} \end{cases} \tag{3.16}$$

$$R_j = \begin{cases} 0 & \text{if } \sum_{i \in L_{n-1}} e_{ij} = 1 \text{ and } \sum_{k \in L_{n+1}} e_{jk} = 1 \\ -\infty & \text{otherwise,} \end{cases} \tag{3.17}$$

The first constraint Q_n in (3.16) means that only one edge among edges between layers $n-1$ and n is chosen. The second constraint R_j in (3.17) means that if node $j \in L_n$ is paired with a node $i \in L_{n-1}$ in the preceding layer ($e_{ij} = 1$), node j must be pared with some $k \in L_{n+1}$ in the next layer ($e_{jk} = 1$). We refer to Q_n and R_j as *uniqueness* and *consistency* constraints, respectively.

Thus, maximizing the likelihood under above constraints leads to the following optimization problem:

$$E^* = \max_{\mathcal{E}} \sum_i \sum_j s_{ij} e_{ij} + \sum_{n=1}^{N} Q_n + \sum_{n=1}^{N} \sum_{j \in L_n} R_j. \tag{3.18}$$

Once the optimal configurations of edges E^* is found, let $\mathcal{I}^*$ denote the set of chosen nodes:

$$\mathcal{I}^*(E^*) = \left\{ i | e_{ij}^* = 1, e_{ij}^* \in E^* \right\}. \tag{3.19}$$

[1]The same is true for binary variable version [45] of the affinity propagation [38].

Similarly to (3.12), the estimate of the source is the mean of the product of Gaussians corresponding $\mathcal{J}^*$:

$$\hat{S} = \left(\sum_{i \in \mathcal{J}^*} \frac{\mu_i}{v_i} \right) \left(\sum_{i \in \mathcal{J}^*} \frac{1}{v_i} \right)^{-1}. \tag{3.20}$$

3.4.2 Belief propagation on the layered graph

We propose a belief propagation (BP) algorithm to infer the optimal configuration of binary variables in (3.18) given noisy measurements. Specifically, we adopt a framework of affinity propagation (AP) [38, 45], which is a variant of BP using pairwise similarities. Given a data set, AP provides as solution a subset that maximizes the total similarity of the chosen configuration. In contrast to original AP where any node can be connected to any other nodes, we allow edge between consecutive layers only (Figure 3.2A). Thus, the proposed algorithm can be understood as a variant of AP with additional layer structure. Hence, we refer to the proposed algorithm as layered affinity propagation (LAP). For our problem, the pairwise similarity in (3.15) is the maximum likelihood of two chosen nodes. Constraints in (3.16) and (3.17) are added so that only one node per layer is chosen as a valid configuration.

Figure 3.2B shows the factor representation of the graph in Figure 3.2A. An edge in Figure 3.2A generates a variable node, shown as a circle in Figure 3.2B. Constraints in (3.16) and (3.17) correspond check nodes, shown as rectangles in Figure 3.2B.

Belief and message on the factor graph are defined as follows. On

each node, the (log) belief of the variable e_{ij} being one or zero are calculated, difference of which defines the outgoing message from the node:

$$\alpha_{ij} = \alpha_{ij}(1) - \alpha_{ij}(0) \tag{3.21}$$

$$\beta_{ij} = \beta_{ij}(1) - \beta_{ij}(0) \tag{3.22}$$

$$\eta_{ij} = \eta_{ij}(1) - \eta_{ij}(0) \tag{3.23}$$

$$\rho_{ij} = \rho_{ij}(1) - \rho_{ij}(0) \tag{3.24}$$

A positive message means that corresponding e_{ij} is more likely to be 1 than 0 given messages from neighbors. A negative message means the opposite.

The messages defined in (3.21-3.24) are updated according to the max-sum rule [71] as follows.

$$\beta_{ij} = s_{ij} + \alpha_{ij} \tag{3.25}$$

$$\rho_{ij} = s_{ij} + \eta_{ij} \tag{3.26}$$

$$\eta_{ij} = - \max_{\{(u,v):u\in L_{n-1},v\in L_n,(u,v)\neq(i,j)\}} \beta_{uv} \tag{3.27}$$

$$\alpha_{jk} = \max_{i\in L_{(n-1)}} \rho_{ij}, \tag{3.28}$$

where $i \in L_{n-1}, j \in L_n, k \in L_{n+1}$ index nodes in three consecutive layers. Note that update rules (3.25)-(3.27) are calculated within a pair of layers while (3.28) involves two pairs of layers (L_{n-1}, L_n) and (L_n, L_{n+1}). Messages from variable to factor nodes in (3.25) and (3.26) directly follow from the

max-sum rule [71] while messages from factor to variable nodes deserve more explanation as follows.

The message α_{ij} from Q_n to e_{ij} in (3.27) is given by the following observation. If $e_{ij} = 1$, all the c_{uv}'s connected to Q_n other than e_{ij} must be zero. Consequently, the message is:

$$\eta_{ij}(1) = \sum_{\{(u,v):u\in L_{n-1},l\in L_n,(u,v)\neq(i,j)\}} \beta_{uv}(0). \tag{3.29}$$

On the other hand, if $e_{ij} = 0$, only one among c_{uv}'s connected to Q_n other than e_{ij} is one and all the remaining must be zero. Therefore, the corresponding message contains the maximum operation as follows:

$$\eta_{ij}(0) = \max_{\{(u,v):u\in L_{n-1},v\in L_n,(u,v)\neq(i,j)\}} \left(\beta_{uv}(1) + \sum_{\{(u',v'):u'\in L_{n-1},v'\in L_n,(u',v')\neq(i,j),(u',v')\neq(u,v)\}} \beta_{u'v'}(0) \right). \tag{3.30}$$

From (3.29) and (3.30), we have the combined message $\eta_{ij} = \eta_{ij}(1) - \eta_{ij}(0)$ in (3.27).

Next, The message α_{jk} from R_j to e_{jk} is given as follows. Suppose $e_{jk} = 1$, meaning that node j in L_n is paired with node in the next layer L_{n+1}. Then, the j must be paired by some node i in the previous layer L_{n-1}. Therefore,

$$\alpha_{jk}(1) = \max_{i\in L_{n-1}} \left(\rho_{ij}(1) + \sum_{l\in L_{n-1},l\neq i} \rho_{ij}(0) \right). \tag{3.31}$$

On the other hand, if $e_{jk} = 0$, j must not be paired with any of nodes in the previous layer.

$$\alpha_{jk}(0) = \sum_{i\in L_{n-1}} \rho_{ij}(0). \tag{3.32}$$

From (3.31) and (3.32), we have the combined message $\alpha_{jk} = \alpha_{jk}(1) - \alpha_{jk}(0)$ in (3.28).

To simplify the algorithm, messages corresponding to constraint R_j is sent to only one direction (from layer n to $n+1$) even though the constraint

itself does not imply any directionality. Initial messages α_{ij}'s to layer 1 are set to zero, corresponding to equal prior. These messages are combined in layer 1 and then new messages are generated and sent to layer 2. In this manner, messages in a layer trigger messages to the next layer. This flow of information is closed by connecting the last layer to the first layer. In other words, outgoing messages from layer N is fed back to layer 1. The directionality and circular boundary condition on layer allow us to simplify the convergence analysis in the following section.

3.4.3 Simplified message updating rules of LAP

According to the max-sum algorithm [71], a node can send a message to its neighboring node only after receiving messages from all the other neighbors. Applying this rule to the structure shown in Figure 3.2B, node e_{ij} sends message β_{ij} as soon as it receives s_{ij} (which is pre-calculated before the first iteration) and α_{ij} (which comes from the previous layer) using (3.25). Therefore, given messages α_{ij} from the previous layer, all β_{ij} with $i \in L_{n-1}$ and $j \in L_n$ are calculated and transmitted to Q_n. Then, Q_n calculates η_{ij} according to (3.27). At this point, e_{ij} receives all the incoming messages and calculates ρ_{ij} using (3.26) and sends it to R_j. Finally, R_j uses ρ_{ij}'s with $j \in L_n$ to produce α_{jk}'s and transmit them to the next layer.

Therefore, massage passing rules involving four messages in (3.25)-(3.28) are combined to those involving only two messages. Substituting (3.25) to (3.27),

$$\eta_{ij} = - \max_{\{(u,v):u\in L_{n-1},v\in L_n,(u,v)\neq(i,j)\}} (s_{uv} + \alpha_{uv}) \tag{3.33}$$

Substituting (3.33) to (3.26),

$$\rho_{ij} = s_{ij} - \max_{\{(u,v):u\in L_{n-1},v\in L_n,(u,v)\neq(i,j)\}} (s_{uv} + \alpha_{uv}) \,. \tag{3.34}$$

Thus, the simplified update rules involve only α_{ij} and ρ_{ij} according to (3.28) and (3.34). Given α_{ij}'s from the previous layer, ρ_{ij}'s are calculated. From ρ_{ij}'s, α_{jk}'s for the next layer are generated and sent through factor node R_j. This procedure is summarized in Algorithm 1.

Algorithm 1 Layered affinity propagation (LAP)

calculate pairwise similarities s_{ij}
initialize $\alpha_{ij} \leftarrow 0$
repeat
  for layer $n = 1, 2, \ldots N$ **do**
    $\rho_{ij} \leftarrow s_{ij} - \max_{\{(u,v):u\in L_{n-1}, v\in L_n, (u,v)\neq(i,j)\}} (s_{uv} + \alpha_{uv})$ for $i \in L_{n-1}, j \in L_n$
    $\alpha_{jk} \leftarrow \max_{i\in L_{n-1}} \rho_{ij}$ for $j \in L_n, k \in L_{n+1}$
  end for
until convergence

3.5 Optimality and computational complexity of LAP

3.5.1 Convergence and correctness of LAP

Now, we study theoretical guarantees of LAP. Specifically, LAP finds the correct MAP solution of (3.18) which is an approximation of the ML estimate in (3.11).

Proposition 1. *The fixed point of LAP is the correct solution of the optimization problem in (3.18).*

In what follows, we use computation tree [75] to prove Proposition 1. The computation tree $\tilde{G}$ of the original graph G is a tree that preserves the local connectivity of G. Nodes $\tilde{e}_{ij}$ and $\tilde{R}_j$ in $\tilde{G}$ are replica of e_{ij} and R_j in G, respectively. Any node in $\tilde{G}$ has as children all the replica of nodes that are neighbors in G except its parent. Figure 3.3 illustrates the computation tree $\tilde{G}$ of G. The root of $\tilde{G}$, denoted as $\tilde{R}_o$, is connected to *all* $\tilde{e}_{ij}, i \in L_{N-1}, j \in L_N$ in layer N. Each $\tilde{e}_{ij}$ is connected to all the other nodes $\tilde{e}_{uv}, u \in L_{N-1}, v \in L_N, (u,v) \neq (i,j)$ through check nodes $\tilde{Q}_N$. Each $\tilde{e}_{uv}$ is connected to the next layer via check node $\tilde{R}_u, u \in L_{N-1}$. In this manner, the computation tree expands from layer N to layer 1 in each iteration. Thus, after T iterations, the depth of $\tilde{G}$ is $4NT$.

Message passing rules for the computation tree is the same as the original tree. By construction, messages on $\tilde{G}$ flow from child to parent. Initially, all the $\tilde{\alpha}_{uv}$ at the leaves of $\tilde{G}$ are set to zero. Once $\tilde{\alpha}_{uv}$ and $\tilde{s}_{uv}$ are given to layer n, messages $\tilde{\rho}_{ij}$ are calculated according to (3.34), which, in turn, produce messages α_{jk} to the next layer according to (3.28).

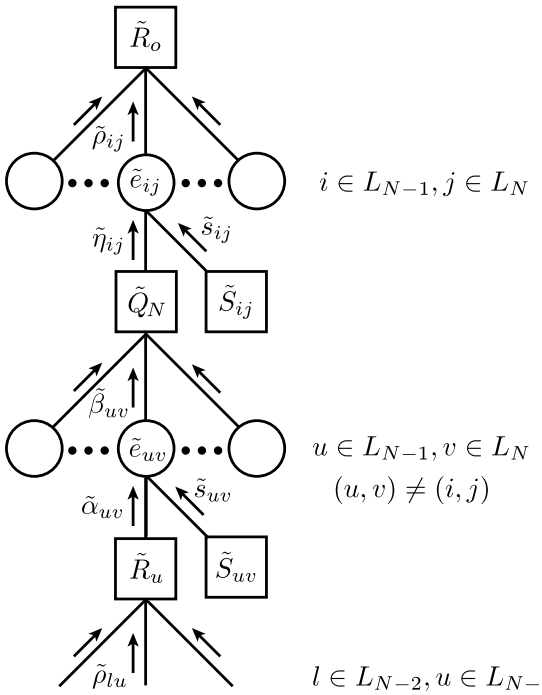

Figure 3.3: The computation tree of the factor graph in Figure 3.2B.

LAP is guaranteed to converge to the correct MAP solution on the computation tree $\tilde{G}$. Let $\tilde{E}^*$ denote a fixed point of LAP on $\tilde{G}$. Previous studies show that a max-sum algorithm always converges to a fixed point, which is the correct MAP configuration on a tree [71]. Because LAP is a max-sum algorithm and $\tilde{G}$ is a tree, $\tilde{E}^*$ is the optimal on $\tilde{G}$. Still, we need to show that optimal configuration $\tilde{E}^*$ on $\tilde{G}$ indeed corresponds to the correct ML estimate in G, which is addressed as follows.

Before discussing optimality, mapping from configuration $\{\tilde{e}_{ij}\}$ on $\tilde{G}$ to $\{e_{ij}\}$ on G is defined as the following. The value of $\tilde{e}_{ij}$ on $\tilde{G}$ is determined by the sign of corresponding $\tilde{\rho}_{ij}$:

$$\tilde{e}_{ij} = \begin{cases} 1 & \text{if } \tilde{\rho}_{ij} > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{3.35}$$

Given $\{\tilde{e}_{ij}\}$ on $\tilde{G}$, corresponding $\{e_{ij}\}$ on G is found by recursively tracing from the root to a leaf. From the root $\tilde{R}_o$, $\tilde{e}_{ij}$ with the largest ρ_{ij} is found

and corresponding e_{ij} is set to 1 in layer N. Next candidate is searched for among the children of the chosen $\tilde{e}_{ij}$. In other words, $\tilde{e}_{uv}$ with the largest $\tilde{\beta}$ is set to 1 and $\tilde{R}_u$ below it is used for next search in layer $N-1$. In this way, a sequence of e_{ij} with value 1 is found for the top N layers and the configuration corresponding $\tilde{E}^*$ is denoted as E^*. This mapping naturally satisfies the consistency constraint and we have the following lemma showing that LAP provides a valid configuration on the original graph.

Lemma 1 (Consistency)**.** *Any configuration E on G generated from a fixed point $\tilde{E}^*$ on $\tilde{G}$ contains one and only one node from each layer.*

Next step is to confirm the optimality of E^* on the original graph G. To show this, we use results of the correctness of max-sum algorithm over a graph with arbitrary structure in [96]. One of the key concepts we adopt from [96] is the optimality of max-sum algorithm over a wide range of neighborhood, called *single loops and trees (SLT) neighborhood*. For completeness, we briefly summarize the definition of SLT and optimality as follows.

Definition 1 (SLT neighborhood [96])**.** *A pair of configurations E_1 and E_2 of G are called SLT neighborhood if there exists set of disconnected trees and single loops T such that E_2 is obtained by changing values of E_1 in T.*

Lemma 2 (SLT-neighborhood optimality [96])**.** *If E^* is a fixed point of a max-sum algorithm, the probability of E^* is greater than any SLT neighbor E_T of E^* in G:*

$$P(E^*|\mathbf{Y}) > P(E_T|\mathbf{Y}), \tag{3.36}$$

where $\mathbf{Y}$ represents measurements.

Since LAP is a max-sum algorithm, Lemma 2 leads to the following Corollary.

Corollary 1. *Let E^* denote the fixed point of LAP. Then, E^* is SLT-neighborhood optimal:*

$$P(E^*|\mathbf{Y}) > P(E_T|\mathbf{Y}), \tag{3.37}$$

for any SLT neighbor E_T of E^* in G, where $\mathbf{Y}$ represents measurements.

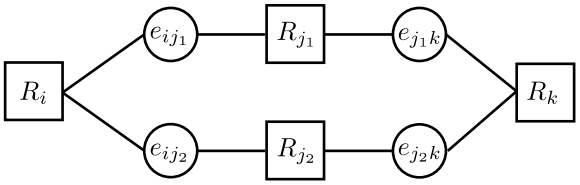

Figure 3.4: E_1 and E_2 are the same configurations except j_1 and j_2 in layer n. By flipping the assignments along the cycle, E_2 is transformed to E_1. This shows that E_1 and E_2 are SLT neighbors.

Finally, SLT-neighborhood optimality of LAP implies global optimality because all the valid configurations are SLT neighborhood to each other. This is shown in the following lemma.

Lemma 3. *Suppose E_1 and E_2 are distinct configurations on G satisfying the consistency. Then, E_1 and E_2 are SLT neighbors to each other.*

Proof. Without loss of generality, one can assume E_1 and E_2 differ in L_n and identical in other layers. Let j_1 and j_2 denote these differing nodes in E_1 and E_2, respectively. Indices of connected nodes in L_{n-1} and L_{n+1} are called i and k, respectively (Figure 3.4). By assigning $e_{ij_1} = 0$, $e_{j_1k} = 0$, $e_{ij_2} = 1$, and $e_{j_2k} = 1$, E_1 is transformed to E_2, which forms a cycle shown in Figure 3.4. Thus, E_1 and E_2 are SLT neighborhood. □

Thus, Lemma 1, Corollary 1, and Lemma 3 prove Proposition 1.

3.5.2 Computational complexity

Now, we compare computational complexities of decoding methods. First, the naïve ML estimation, based on evaluating likelihoods of quantized bins, requires a large number of bins to represents a continuous source and the computational complexity scales with the number of bins. Next, the local-maxima-based approach in (3.11) needs $O(a_{max}^N \log_2 N)$ operations, where $a_{max} = \max_n a_n$ and the term $\log_2 N$ is based on the assumption that N Gaussians are recursively combined using a binary tree for each local max-

imum. Note that above two methods assume a centralized decoder that has access to all the measurements.

Compared to the complexities of these central schemes, LAP's computational complexities in individual nodes are much lower. The numbers of messages sent to nodes R_j and Q_n are less than a_{max} and a_{max}^2, respectively, where $a_{max} = \max_n a_n$. Thus, the computational complexity per node is $O(a_{max}^2)$ while the number of nodes scales as $O(Na_{max}^2)$

3.6 Numerical simulations

Numerical simulations are performed as follows. The number of congruences N are either 5 or 8. The parameter of a_n is designed such that resulting codewords are well separated and produce a low threshold error probability: $A_5 = \{9, 11, 13, 17, 23\}$ and $A_8 = \{11, 13, 15, 17, 23, 29, 37, 47\}$. Readers are referred to [103] for detail analysis and procedure to find such good codes. For a given SNR $= \frac{1/12}{\sigma^2}$, 10^4 set of noisy congruences are generated using (3.3) and (3.4) for source $S = 0.5$. For each set, the proposed algorithm (LAP) and the maximum likelihood (ML) estimation produce estimates S_{LAP} and S_{ML}, respectively. For LAP, the maximum number of iteration is 20. For ML, the source range is quantized into 10^5 bins with equal size. The likelihood is calculated for each bin using Eq. (3.6) and the bin with the maximum likelihood is found as an estimate. The mean square error (MSE) is used as a performance measure.

LAP converges within a few iterations. Figure 3.5 shows an example of LAP when true $S = 0.5$. Circles in Figure 3.5 A and B represent the center of the mixture of Gaussians from each layer. Solid blue lines show the edges assigned with value 1. Dashed red lines indicate local maxima ρ_{ij} in each layer (shown only when they differ from the blue lines). This discrepancy after the first iteration (Figure 3.5A) is resolved as messages propagate over iterations (Figure 3.5B). In Figure 3.5C, the log likelihood of the configuration increases until 4 iterations and then saturates, indicating the convergence of LAP.

The mean square error (MSE) of LAP is essentially the same as the ML estimation for high signal-to-noise ratios (SNRs). Figure 3.6 shows MSEs of LAP (blue circle) and ML (red square) for a range of SNRs. Dashed line is a lower bound of MSE when correct set of Gaussians is revealed to the decoder

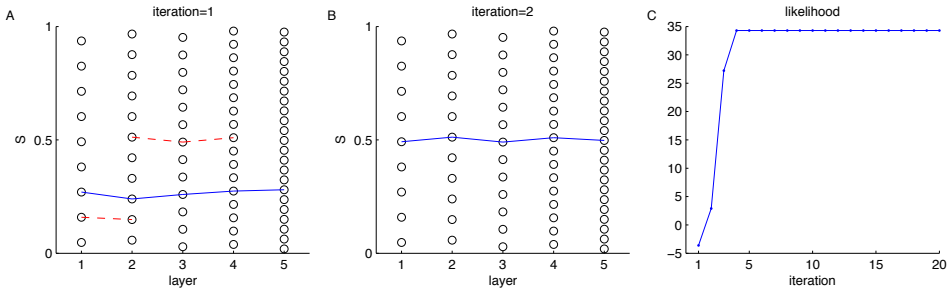

Figure 3.5: An example of layered affinity propagation with $S = 0.5$ and $a_n = \{9, 11, 13, 17, 23\}$. In A and B, circles show the centers of Gaussians ordered by the layer along the x-axis Solid blue lines show chosen edges after 1 and 2 iterations (A and B, respectively). Dashed red lines indicate maximum ρ_{ij} in each layer (shown only when it differs from the black lines). LAP converges to the right solution after 2 iterations In C, the log likelihood continues increasing until 4 iterations and then saturates.

(Appendix B.2). For high SNRs, MSEs of LAP and ML estimation are almost identical. For low SNRs, MSE of LAP deviate from that of ML estimation. Overall, LAP achieves high accuracy with much fewer computations.

3.7 Conclusion

In summary, we propose a belief propagation algorithm to estimate a continuous source from its noisy residues, which is inherently a combinatorial optimization. We first approximate the original estimation with a simplified one involving only pairwise likelihoods and then solve the simplified problem by a belief propagation on the graph with layered structure. The proposed algorithm, called layered affinity propagation (LAP), is guaranteed to find the MAP solution of the approximated problem. Numerical simulations demonstrate excellent convergence within a few iterations and accuracy of the proposed algorithm in estimating the source.

In high SNRs, the proposed algorithm achieves essentially the same mean square error of maximum likelihood estimation. There is a gap for low SNRs, which can be understood as follows. LAP is based on local computations using two congruences while the ML estimation considers all the congruences simultaneously. When noise is large and measurements are close to the decision

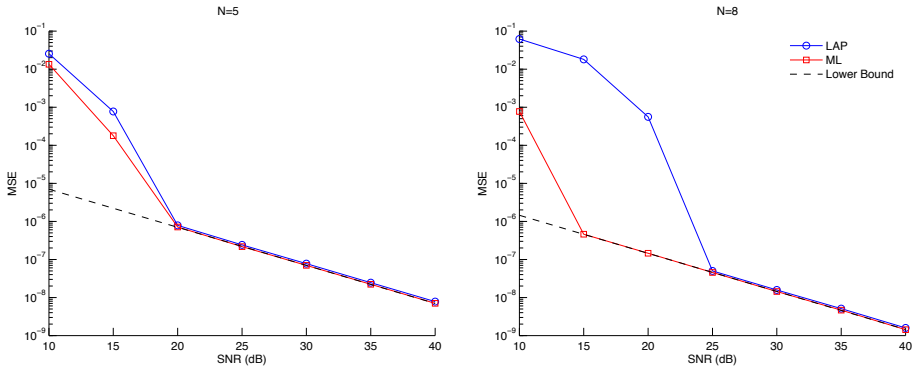

Figure 3.6: The mean square error (MSE) of LAP is essentially the same as that of ML for high SNRs and comparable for low SNRs. For $N = 5$ (left) and 8 (right), $a_n = \{9, 11, 13, 17, 23\}$ and $\{11, 13, 15, 17, 23, 29, 37, 47\}$, respectively. 10^4 set of noisy samples are generated for each SNR and estimated by LAP (blue circle) and by ML estimation (red square). Mean square errors of LAP (blue circle) and of ML estimation (red square) are plotted as function of SNR=$\frac{1/12}{\sigma^2}$. Dashed line shows a lower bound without threshold error. (See Appendix B.2.)

boundary, local information based on only few congruences may lead to an incorrect solution. Understanding this behavior of LAP in low SNRs and designing a way to avoid this limitation would be future research topics.

Finally, another interesting future research is to study plausible neural implementations of the proposed algorithm. We focus only on algorithmic aspect of reconstructing a continuous source from its noisy residues. How the brain addresses this problem would be an intriguing question to be addressed.

Chapter 4

Distributed joint source-channel coding using the brain's multi-scale code

4.1 Introduction

In previous chapters, we discuss on the amount of information in the multi-scale code in the entorhinal cortex, which depends on decoding method, (Chapter 2) and then propose an alternative decoding algorithm that recovers the source based on local computations (Chapter 3). In this chapter, we demonstrate an application of such multi-scale codes for distributed coding.

4.1.1 Analog codes as joint source-channel coding schemes

Analog codes, where both the input and output fields for the code correspond to points on the real line, have gained particular attention in recent years. Such codes differ from more conventional coding ensembles where typically one of the input or output sets is discrete-valued. The analog code maps a continuous source to a continuous codeword without explicit quantization. When this mapping is optimized to be a good source code and, at the same time, a good channel code, this analog code results in a joint source-channel (JSC) code.

Analog codes provide three advantages over conventional discrete coun-

terpart. The first is small delay. Conventional codes, where continuous input is first quantized to discrete codeword by a source code and then encoded by a channel code, are designed to maximize spectral efficiency at the price of large delay due to long codewords. In contrast, analog codes, as JSC codes, are designed to minimize distortion for a fixed code length that are much smaller than that of conventional codes. Thus, analog codes are ideally suited for delay-limited applications as well as for channels with time-varying signal-to-noise ratio (SNR). Second, analog codes are robust to SNR mismatch and offer graceful degradation of the performance. Separate source and channel coding is known to be optimal for a given SNR, but the performance of separation is not robust to SNR mismatch [46]. Analog codes can render a degree of robustness to the system, and hence they are analyzed in certain detail in existing literature [25, 92, 94]. Third, the use of analog code is justified by theories as well. In many cases, it is known that uncoded transmission (a trivial analog code) of continuous source(s) over classes of additive Gaussian noise (AGN) channels outperforms separate source and channel coding [30, 43], and is sometimes optimal [42, 88].

Constructing "good" analog codes has been an intriguing research problem. One of the desired properties of analog code is that codewords are maximally separated from one another. To achieve this goal, various mappings from continuous source to continuous codeword are investigated. Chen and Wornel proposed to use chaotic dynamic systems to generate analog codewords [25]. Among such mappings, the shift-map has received particular attention [92,94].

Space-filling curves are used to design efficient joint source-channel coding schemes when bandwidths of the source and the channel do not match [26]. Another interesting approach is to derive an analog code from a discrete code with maximum distance [76]. In previous studies, analog codes are studied for point-to-point channels. In this paper, we generalize the notion of the shift-map code and apply it to construct a joint source channel code for distributed coding.

4.1.2 Joint source-channel coding for distributed coding

In distributed settings, multiple senders and receivers wish to exchange informations and optimal coding schemes for distributed coding quite differ from those for point-to-point communications. For example, in distributed source coding, correlated sources are encoded by separate senders and transmitted to a receiver. The celebrated result by Slepian and Wolf shows that the separate encoders can be as efficient as the encoders with full cooperation in terms of required data rate [83]. An asymmetric version of distributed source coding is that the receiver is interested only one of the sources and the other source is used as side information to improve estimation accuracy of the desired one. This problem is first studied for lossless [100] and then for lossy source coding [101], which is referred to as Wyner-Ziv source coding problem. Distributed source coding has numerous applications including distributed compression, distributed video coding, multi-view video compression, to name just a few [34].

In order to achieve the best efficiency, two senders should provide complementary informations to the receiver. However, how is this possible without any cooperation between senders? This goal is achieved by a discrete codebook with random binning structure [83]. Sender 1 transmits only the bin index of the codeword to reduce data rate. The receiver first decode this bin index and uses the signal from sender 2 to identify the codeword in the bin. The correlation between the sources allows the receiver to use joint typical decoder for the second step and guarantees small error probabilities [83].

We extend the binning strategy for a discrete codebook to design the continuous codebook which dynamically scales with side information separately provided to the decoder. We show that a generalized version of shift-map codes enables robust joint source-channel coding by separate encoders. To be specific, choosing relatively prime integers for code generation modifies the shift-map code to interleave itself in the coding space. Since this use of relatively prime integers shares a strong connection with redundant residue number systems (RRNS) [54, 84, 95, 102], we refer to our codes as RRNS-map codes.

4.2 System model

Figure 4.1 presents the system model. The continuous source S is uniformly distributed in the unit interval $[0, 1)$. This source is encoded in terms of N real-valued variables $\mathbf{X} = (X_1, X_2, \ldots, X_N)$. This codeword is transmitted over additive Gaussian noise (AGN) channels with the constraint

$0 \leq X_n < 1$ for $n = 1, 2, \ldots, N$[1]. The received signal $\mathbf{Y} = (Y_1, Y_2, \ldots, Y_N)$ is the sum of the transmitted codeword and noise $\mathbf{Z} = (Z_1, Z_2, \ldots, Z_N)$:

$$Y_n = X_n + Z_n, n = 1, 2, \ldots, N \tag{4.1}$$

$$Z_n \sim \mathcal{N}(0, \sigma^2), \tag{4.2}$$

where the noise terms Z_n's are zero-mean Gaussian with variance σ^2 and independent of one another.

The receiver has *side information* that the source S lies in a subinterval $W \subset [0, 1]$. This side information is assumed to be known to the receiver but not to the sender, in a setting analogous to Wyner-Ziv source coding problem [100, 101]. The difference from [100, 101] is that we include AGN channels in the consideration and seek for a joint source-channel coding scheme with side information at the decoder.

For example, the side information may be provided in the following manner. A separate encode observes S' which is correlated with S and transmits S' to the receiver:

$$S' = S + Z_c \tag{4.3}$$

$$Z_c \sim \mathcal{N}(0, \sigma_c^2). \tag{4.4}$$

Then, the receiver estimates S' with distortion D_2. Then, the overall uncertainty of S is $\sigma_c^2 + D_2$ in terms of variance. Thus, we define the side information

[1]With an appropriate offset, this constraint is equivalent to the power constraint $E\left[(X_n)^2\right] \leq \frac{1}{12}$ for $i = 1, 2, \ldots, N$.

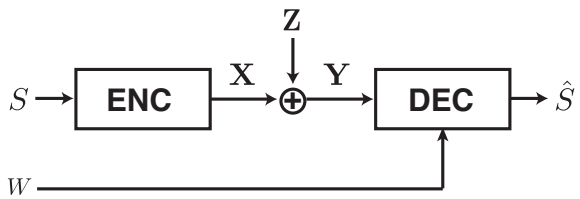

Figure 4.1: The schematic diagram of the system model. The encoder generates codewords X for source $S \in [0, 1)$, which are subsequently transmitted over AGN channels. In addition to the noisy observation Y, the decoder has additional knowledge, *side information*, that the source S lies in an interval $W \subset [0, 1]$.

$W = \left[\hat{S}' - \frac{|W|}{2}, \hat{S}' + \frac{|W|}{2}\right]$, where $\hat{S}'$ is the estimate of $\hat{S}'$. The outage probability is

$$P_o = 2\left(1 - \Phi\left(\frac{|W|}{2(\sigma_c + \sqrt{D_2})}\right)\right), \tag{4.5}$$

where $\Phi(x)$ is the cumulative distribution of the standard normal distribution defined by $\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} dt$. For simplicity, we assume that this outage probability is negligible with an appropriate choice of $|W|$.

The receiver uses this side information W and the channel output $\mathbf{Y}$ to produce an estimate of S, denoted as $\hat{S}$. The distortion D is defined as the mean square error in the estimation of S:

$$D = E\left[\left(\hat{S} - S\right)^2\right]. \tag{4.6}$$

The distortion generally has two distinct components. The first is due to codewords that are mapped by noise to other codewords that represent nearby points in the source (Figure 4.2 blue). These errors occur frequently but the

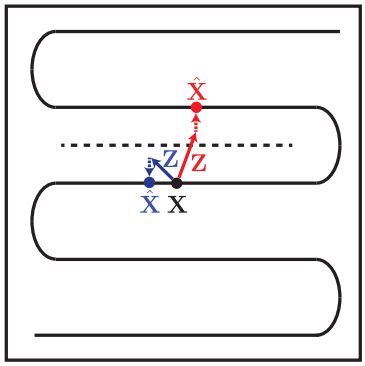

Figure 4.2: The notion of threshold error in dimension-expansion mappings. A source S of dimension 1 (e.g. the unit interval) is mapped to a codeword $\mathbf{X}$ of higher dimension. This mapping may be viewed as an embedding of a curve in a higher dimensional space. The length of the embedded curve is L and distant segments of the embedded curve are separated by a minimum distance d. A small noise (blue) is decoded as a point on the correct line segment, and the resulting error is small (local error). In contrast, a large noise (red) is decoded to the adjacent segment and is the estimation error is large (threshold error).

magnitudes are small. The second is due to noise that maps a codeword to another that represents a distant point in the source (Figure 4.2 red). These errors, called *threshold errors* by Shannon [81], are rare if the noise variance is small, but magnitudes are large. By considering the probability of each case and the corresponding mean square error, the distortion D in (4.6) is:

$$\begin{aligned} D &= P(\mathbf{Z} \notin \mathcal{T}) E\left[(\hat{S}-S)^2 | \mathbf{Z} \notin \mathcal{T}\right] \\ &\quad + P(\mathbf{Z} \in \mathcal{T}) E\left[(\hat{S}-S)^2 | \mathbf{Z} \in \mathcal{T}\right], \end{aligned} \tag{4.7}$$

where $\mathcal{T}$ represents the set of noise vectors that produce threshold errors.

4.3 The shift-map code and its generalization

We first start with the shift-map code which is studied in the context of point-to-point communications and then generalize the shift-map code to a new construction for distributed coding.

4.3.1 The shift-map code

Given a continuous source S, the shift-map codeword of length N is defined as follows [25, 92, 94]:

Definition 2 (The shift-map code)**.** *For a source $S \in [0,1)$ and the design parameter α, the shift-map codeword $\mathbf{X}^{SM}(S)$ of length N is defined as*

$$\mathbf{X}^{SM}(S) = [X_1^{SM}(S), X_2^{SM}(S), \ldots, X_N^{SM}(S)] \tag{4.8}$$

$$X_n^{SM}(S) = \alpha^{(n-1)} S \bmod 1, \ (n = 1, 2, \ldots, N), \tag{4.9}$$

where $\alpha \geq 2$ is a positive integer that controls scaling of the codeword. The set of all the codewords is denoted as $\mathcal{X}^{SM}$ and the set of codewords corresponding to source in the interval $W \subset [0,1]$ $\mathcal{X}^{SM}(W)$:

$$\mathcal{X}^{SM} = \left\{\mathbf{X}^{SM}(S) | S \in [0,1)\right\} \tag{4.10}$$

$$\mathcal{X}^{SM}(W) = \left\{\mathbf{X}^{SM}(S) | S \in W\right\}. \tag{4.11}$$

The shift-map codebook $\mathcal{X}^{SM}$ forms parallel line segments with direction $(1, \alpha, \alpha^2, \ldots, \alpha^{N-1})$ inside the unit hypercube (Figure 4.3A). The total arc-lengh over all these segments is called the stretch factor, denoted as $L^{SM}(\alpha)$, and the minimum distance between them $d^{SM}(\alpha)$.

For a fixed N, there exists a trade-off between the stretch factor and the minimum distance, controlled by α. Specifically, $L^{SM}(\alpha)$ monotonically increases while $d^{SM}(\alpha)$ monotonically decreases with increasing α. To be specific, we have the following lemma.

Lemma 4. *Given the shift-map code with α, the stretch factor $L^{SM}(\alpha)$ and the minimum distance $d^{SM}(\alpha)$ are as follows:*

$$L^{SM}(\alpha) = \sqrt{1+\alpha^2+\cdots+\alpha^{2(N-1)}} \tag{4.12}$$

$$d^{SM}(\alpha) = \frac{1}{\alpha}\frac{\sqrt{(L^{SM}(\alpha))^2-1}}{L^{SM}(\alpha)}. \tag{4.13}$$

Proof. The stretch factor in (4.12) is because the message interval of length 1 is *stretched* by α^{n-1} in the n'th dimension for each $n = 1, 2, \ldots N$. In order to calculate the minimum distance in (4.13), let us first consider distances from individual line segments to a reference line passing the origin. Finding intercepts of individual line segments simplifies the calculation as follows. Consider a non-zero codeword $\mathbf{X}$ with the n_o'th coordinate being zero for $1 < n_o \leq N$. By construction, $X_{n_o} = 0$ implies $X_n = 0$ for $n > n_o$. These intercepts correspond to the message $S = \frac{i}{\alpha^{(n_o-1)}}$ for a positive integer $0 < i < a_n$. Thus, other coordinates have the form of $X_n = \frac{\alpha^{(n-1)}i}{\alpha^{(n_o-1)}}$ for $n < n_o$. Among those points, $(\frac{1}{\alpha}, 0, \ldots, 0)$ attains the minimum distance in (4.13). □

For large α, the minimum distance scales inversely with α: $d \approx \frac{1}{\alpha}$. To be specific, we have the following corollary.

Corollary 2.

$$\frac{4}{5}\frac{1}{\alpha^2} \leq d^{SM}(\alpha)^2 < \frac{1}{\alpha^2}. \tag{4.14}$$

Proof. Equation (4.14) immediately follows from (4.13) by observing that

$$\frac{4}{5} \leq \frac{(L^{SM}(\alpha))^2 - 1}{(L^{SM}(\alpha))^2} = 1 - \frac{1}{(L^{SM}(\alpha))^2} < 1 \tag{4.15}$$

since $(L^{SM}(\alpha))^2 \geq 5$ for $\alpha \geq 2$. □

The tradeoff between the stretch factor and the minimum distance results in the trade-off between two terms in the distortion in (4.7). For a given dimension N and a fixed σ, increasing $L^{SM}(\alpha)$ by increasing α reduces the size of distortions provided that there is no threshold error. Specifically, the first term in (4.7), scales as:

$$E\left[(\hat{S} - S)^2 | \mathbf{Z} \notin \mathcal{T}\right] \propto \frac{1}{L^2} \tag{4.16}$$

However, increasing the stretching factor means that a longer curve is packed into a fixed volume. Therefore, the minimum distance $d^{SM}(\alpha)$ decreases and the probability of threshold errors increases. As a result, the second term in the distortion of (4.7) grows with increasing α.

4.3.2 A generalization of the shift-map code to use side information at the decoder

The shift-map code is generalized by relaxing the conditions on the scaling coefficients in (4.9). In the shift-map code, the source is multiplied by

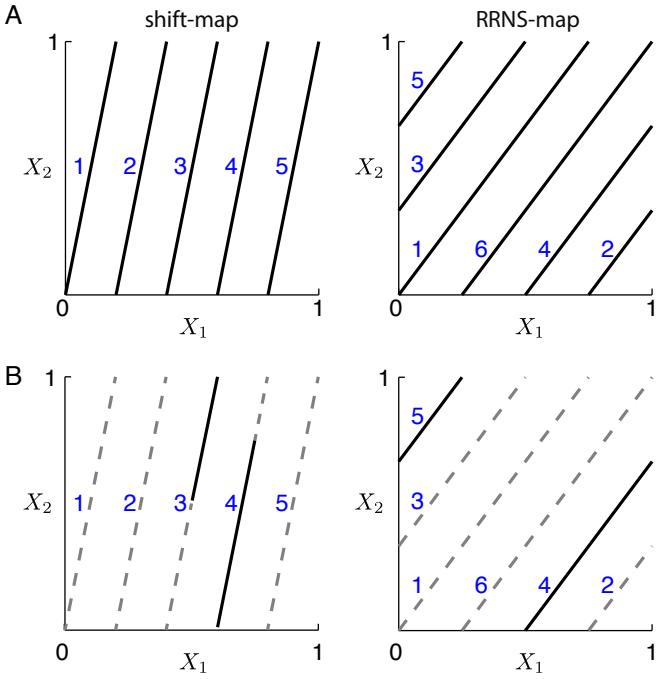

Figure 4.3: Examples of the shift map with $(a_1, a_2) = (1, 5)$ on the left and the *RRNS-map* with $(a_1, a_2) = (3, 4)$ on the right. Both codes are chosen to have similar stretch factors and minimum distances. Numbers next to individual segments in blue indicate the order of the mapping as the source increases from 0 to 1. A and B show codebooks without and with side information $W = [0.5, 0.75]$, respectively. Gray dashed lines correspond the codeword inconsistent with W.

a geometric series followed by the modulo operation. In contrast, we propose to use relatively prime numbers to encode the source with its residues modulo using relatively prime numbers This representation is, in principle, closely related to the redundant residue number system (RRNS) codes [84] in which an integer is encoded by their residues with respect to a set of relatively prime moduli. Due to this strong connection, we refer to the proposed code as the RRNS-map code.

Specifically, the RRNS-map code is defined as follows:

Definition 3 (The RRNS-map code)**.** *For a source $S \in [0,1)$ and the design parameter $\mathbf{a} = (a_1, a_2, \ldots, a_N)$, the RRNS-map codeword $\mathbf{X}^{RRNS}(S)$ is defined as:*

$$\mathbf{X}^{RRNS}(S) = [X_1^{RRNS}(S), X_2^{RRNS}(S), \ldots, X_N^{RRNS}(S)] \tag{4.17}$$

$$X_n^{RRNS}(S) = a_n S \mod 1, \quad (n = 1, 2, \ldots, N) \tag{4.18}$$

$$gcd(a_n, a_m) = 1 \quad \textit{for } n \neq m, \tag{4.19}$$

where the scaling coefficients $1 < a_1 < \cdots < a_N$ are positive integers and $gcd(\cdot,\cdot)$ represents the greatest common divisor of the two numbers. The set of all the codewords generated by the RRNS code is denoted as $\mathcal{X}^{RRNS}$ and the partial codebook for $S \in W \subset [0,1)$ $\mathcal{X}^{RRNS}(W)$:

$$\mathcal{X}^{RRNS} = \left\{\mathbf{X}^{RRNS}(S) | S \in [0,1)\right\} \tag{4.20}$$

$$\mathcal{X}^{RRNS}(W) = \left\{\mathbf{X}^{RRNS}(S) | S \in W\right\}. \tag{4.21}$$

The RRNS-map codebok $\mathcal{X}^{RRNS}$ forms parallel line segments with direction $(a_1, a_2, \ldots, a_N)$ inside the unit hypercube (Figure 4.3B). The stretch factor $L^{RRNS}(\mathbf{a})$ and the minimum distance $d^{RRNS}(\mathbf{a})$ are defined as the same as for the shift-map code.

The key difference between the shift-map and RRNS-map codes is the choice of the parameter, which leads to the following differences. In the shift-map code, the n'th coordinate of the codeword is related to the source S by

a scale factor $\alpha^{(n-1)}$, which increases with n. Therefore, the coordinate with larger n encodes local changes of the source with larger sensitivity. In contrast, in RRNS-map codes, one can choose a_n's within a small range so that all the X_n's have similar sensitivities to the source.

For instance, Figure 4.3A compares examples of shift-map (left) and RRNS-map (right) codes. The parameters for those codes are $(a_1, a_2) = (1, 5)$ and $(a_1, a_2) = (3, 4)$, respectively. Both codes have essentially the same stretch factor ($L^2 = 1+5^2 \approx 3^2+4^2$) and minimum distance (0.20). Therefore, without additional information about the source, decoders for both codes have the same performance. The numbers in blue indicate the order of the encoding line segments, as the source ranges from 0 to 1. In the conventional shift-map code, each segment of codeword monotonically increases, and the resulting segments are ordered sequentially. However, in the RRNS-map code, the order may be interleaved. This interleaving structure is a key advantage of the RRNS-map codes over conventional shift-map codes when combined with side information.

In Figure 4.3B, the effects of side information on the shift-map (left) and RRNS-map (right) codes are demonstrated. Solid lines indicate codewords consistent with side information $W = [0.5, 0.75]$, corresponding to segments 3 and 4 for the shift-map code and segments 4 and 5 for the RRNS-map code. Gray dashed lines shows codewords inconsistent with W. In the shift-map codes, the distance between candidate segments remains the same because they lie next to each other. In contrast, the distance between candidate segments is much larger in the RRNS-map code because they are non-consecutive,

resulting in a larger minimum distance and, consequently, a lower threshold error.

Thus, the design goal for the RRNS-map code is two-fold. The first is to achieve well-spaced segments with the same or approximately the same stretch factor and minimum distance as the corresponding shift-map code, which guarantees the same distortion in the absence of side information. The second is to interleave the order of the coding segments so that neighboring segments encode distant subintervals of the source. When this interleaving property is combined with side information, the effective minimum distance between coding segments increases without a decrease in local stretch factor, and the overall result is a smaller distortion without changing encoding scheme.

4.4 RRNS-map codes without side information

Before considering side information at the decoder, let us first study the structure of the RRNS-map codebook in the absence of side information.

4.4.1 Geometric interpretation: "Cylinder packing" problem

Let us discuss on finding a good RRNS-map with maximum separation in a geometrical perspective. A cylinder around the RRNS-map codebook with radius r is defined as follows:

$$\mathcal{C}(r) = \left\{ \mathbf{x} \in [0,1)^N : |\mathbf{x} - \mathbf{x}^{RRNS}| < r,\ \forall \mathbf{x}^{RRNS} \in \mathcal{X}^{RRNS} \right\}, \qquad (4.22)$$

where $|\cdot|$ is the Euclidean distance. Then, the problem is to find the direction $\mathbf{a} = (a_1, a_2, \ldots, a_N)$ of the cylinder with the maximum radius r under the constraint that $C(r)$ does not intersect itself. We call this problem as *cylinder packing*. Since $\mathbf{a}$ consists of relatively prime integers, the cylinder packing problem is an integer programming with unusual search domain, denoted as the following:

$$\begin{aligned} \mathcal{A} &= \{(a_1, a_2, \ldots, a_N) \mid \gcd(a_n, a_m) = 1 \text{ if } n \neq m, \\ &\qquad a_n < a_m \text{ if } n < m\} \end{aligned} \tag{4.23}$$

The requirement of relatively prime coordinates might appear to be too stringent and one might think that the size of the search space $\mathcal{A}$ itself could be small. By contrast, as the dimension N grows, the probability of finding points with relatively prime coordinates among the N-dimensional rectangular lattice quickly approaches to 1 [6], meaning that there are many candidate RRNS-map codes. To simplify the problem, we restrict the search domain to those vectors in $\mathcal{A}$ with coordinates that are not greater than a predetermined a_{max}, denoted as

$$\begin{aligned} \mathcal{A}_{a_{max}} &= \{(a_1, a_2, \ldots, a_N) \mid \gcd(a_n, a_m) = 1 \text{ if } n \neq m, \\ &\qquad a_n < a_m \text{ if } n < m,\ a_n \leq a_{max}\} \end{aligned} \tag{4.24}$$

This restriction avoids too large stretch factors that produce severe threshold errors and large distortions.

In order to find the tightest cylinder packing, let us consider the hyperplane that is orthogonal to the cylinder axis. The hyperplane that is orthogonal

to $\mathbf{a}$ and passes through the center of the unit hypercube is denoted by $H(\mathbf{a})$. Algebraically, this hyperplane is

$$H(\mathbf{a}) = \left\{\mathbf{x} \in [0,1)^N \mid \mathbf{a} \cdot (\mathbf{x} - \mathbf{c}) = 0\right\}, \tag{4.25}$$

where $\mathbf{c} = \left(\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2}\right)$ is the center of the unit hypercube and $\cdot$ represents the inner product. Then, each line segment in $\mathcal{X}^{RRNS}$ intersects with $H(\mathbf{a})$ at a point. We call the set of such intersections $\mathcal{X}^*$ *discrete codebook*, denoted as:

$$\begin{aligned} \mathcal{X}^* &= H(\mathbf{a}) \cap \mathcal{X}^{RRNS} && (4.26) \\ &= \{\mathbf{a}S \bmod 1 \mid S \in [0,1), \mathbf{a} \cdot (\mathbf{a}S \mod 1) = \mathbf{a} \cdot \mathbf{c}\}. && (4.27) \end{aligned}$$

Similarly, the discrete codebook corresponding source in the interval W is defined as follows:

$$\begin{aligned} \mathcal{X}^*(W) &= H(\mathbf{a}) \cap \mathcal{X}^{RRNS}(W) && (4.28) \\ &= \{\mathbf{a}S \bmod 1 \mid S \in W, \mathbf{a} \cdot (\mathbf{a}S \mod 1) = \mathbf{a} \cdot \mathbf{c}\}. && (4.29) \end{aligned}$$

Maximizing the radius of the cylinder is equivalent to maximizing the distance between points in the discrete codebook. The minimum distance of the RRNS-map code is equal to the minimum distance between distinct points in $\mathcal{X}^*$:

$$d^{RRNS}(\mathbf{a}) = \min_{x \neq x' \in \mathcal{X}^*} |\mathbf{x} - \mathbf{x}'| \tag{4.30}$$

Unfortunately, the minimum distance depends on $\mathbf{a}$ in a non-trivial way and the search domain $\mathcal{A}$ is not convex. In the following subsection, we provide a way to identify all the points in $\mathcal{X}^*$ for a given $\mathbf{a}$ and to calculate the minimum distance $d^{RRNS}(\mathbf{a})$.

4.4.2 Identifying the discrete codebook

For a given direction $\mathbf{a}$, points in the discrete codebook $\mathcal{X}^*$ are calculated as follows. We first consider the intersections of $\mathcal{X}^{RRNS}$ and the faces of the unit cube and then project those intercepts onto H. Let H_n be the hyperplane that is orthogonal to the n'th axis and contains the origin:

$$H_n = \left\{ \mathbf{x} \in \mathbb{R}^N \mid \mathbf{e_n} \cdot \mathbf{x} = 0 \right\}, \tag{4.31}$$

where $\mathbf{e_n}$ is the unit vector with zero in all the coordinates other than the n'th coordinate. Then, $\mathcal{X}^{RRNS}$ intersects with the face of the unit hypercube on H_n at the origin and the other $(a_n - 1)$ points, as summarized in the following lemma.

Lemma 5. *$\mathcal{X}^{RRNS}$ intersects with $H_n(\mathbf{a})$ at a_n number of points with corresponding source $S = \frac{i}{a_n}, i = 0, 1, \ldots, a_n - 1$: Let*

$$\begin{aligned} \mathcal{X}_n(\mathbf{a}) &\equiv H_n \cap \mathcal{X}^{RRNS} && (4.32) \\ &= \left\{ \mathbf{a}S \ mod\ 1 \;\middle|\; S = \frac{i}{a_n}, i = 0, 1, \ldots, a_n - 1 \right\}. && (4.33) \end{aligned}$$

Then,

$$|\mathcal{X}_n(\mathbf{a})| = a_n \tag{4.34}$$

Proof. By the definition, $\mathcal{X}_n$ is the set of points in $\mathcal{X}$ with the n'th coordinate being zero, which is equivalent to

$$a_n S = 0 \pmod 1. \tag{4.35}$$

Since $0 \leq S < 1$, $0 \leq a_n S < a_n$. Thus, (4.35) has a_n solutions of S, namely $S = \frac{i}{a_n}, i = 0, 1, \ldots, a_n - 1$. □

Next, the primality of a_n's in the construction of the RRNS-map code implies that the projections of $\mathcal{X}_n \setminus \mathbf{0}$ onto H are disjoint as stated in the following lemma.

Lemma 6. *If* $n \neq m$,

$$\mathcal{X}_n(\mathbf{a}) \cap \mathcal{X}_m(\mathbf{a}) = \mathbf{0}, \tag{4.36}$$

where $\mathbf{0}$ *represents the vector corresponding to the origin* $(0, 0, \ldots, 0)$.

Proof. Trivially, $\mathbf{0} \in X_n$ for all $n = 1, 2, \ldots, N$. Next, it is shown by contradiction that there is no other element in the intersection. Suppose that there exists an element other than $\mathbf{0}$ in the intersection: $\mathbf{x} \in \mathcal{X}_n \cap \mathcal{X}_j$ and $\mathbf{x} \neq \mathbf{0}$. This implies that there exists $s \in (0, 1)$ satisfies (4.35) for both a_n and a_m. In other words, $a_n S = m$ and $a_m S = n$ with integers $0 < m < a_n - 1$ and $0 < n < a_m - 1$. Since a_n and a_m are relatively prime, this can happen only when $S = 0$, which contradicts the assumptions that $\mathbf{x} \neq \mathbf{0}$. □

Lemma 5 and 6 lead to the following theorem showing that the number of points in $\mathcal{X}^*$ is related to the sum of a_n's.

Theorem 1. *Given a RRNS-map code with* $\mathbf{a}$ *satisfying (4.19), the cardinality of the discrete codebook* $\mathcal{X}^*$*, the intersections between the codebook* $\mathcal{X}^{RRNS}$ *and*

the orthogonal hyperplane H in (4.25), is as follows:

$$|\mathcal{X}^*| \;=\; 1+\sum_{n=1}^{N}(a_n-1)\,, \tag{4.37}$$

where $|\mathcal{X}^*|$ represents the cardinality of $\mathcal{X}^*$.

Following the same procedure, one can identify all the points in $\mathcal{X}^*$ by first finding $\mathcal{X}_n(\mathbf{a})$ and then projecting those points and the origin onto H. Let $\mathbf{x}_{ni}$ be an element in $X_n(\mathbf{a})$ with corresponding source $S=\frac{i}{a_n}$, $i\neq 0$ from (4.33):

$$\mathbf{x}_{ni}=\left(\frac{a_1}{a_n}i,\ldots,\frac{a_{n-1}}{a_n}i,0,\frac{a_{n+1}}{a_n}i,\ldots,\frac{a_N}{a_n}j\right)\bmod 1, \tag{4.38}$$

where $n=1,2,\ldots,N$ and $i=1,\ldots a_n-1$. In order to project this point onto H, one needs to find the orthogonal basis of H by finding the null space of $\mathbf{a}$. Considering $\mathbf{a}^T$ as a $(1\times N)$ matrix and performing the singular value decomposition,

$$\mathbf{a}^T=\begin{bmatrix}a_1 & a_2 & \ldots & a_N\end{bmatrix}=\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T. \tag{4.39}$$

$\mathbf{\Sigma}$ contains only one non-zero singular value at the first row and all the other singular values are zero. Thus, the first column of $\mathbf{V}$ corresponds to the non-zero singular value and the remaining $(N-1)$ columns of $\mathbf{V}$ are the orthogonal basis of the null space of $\mathbf{a}$, denoted as a $N\times(N-1)$ matrix $\mathbf{B}$. Therefore, the projection of $\mathbf{x}_{ni}$ onto H is

$$\mathbf{x}^*_{ni}=\mathbf{B}^T(\mathbf{x}_{ni}-\mathbf{c}),\ n=1,2,\ldots,N, i=1,2,\ldots,a_n-1 \tag{4.40}$$

where $\mathbf{c} = \left(\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2}\right)$ is the center of the hypercube. From (4.40), we have the full description of individual points in $\mathcal{X}^*$, which allows us to numerically calculate the minimum distance for a given $\mathbf{a}$.

4.4.3 The trade-off between the minimum distance and the stretch factor

Similarly to shift-map codes, there exists the fundamental trade-off between the minimum distance $d^{RRNS}(\mathbf{a})$ and the stretch factor $L^{RRNS}(\mathbf{a})$ of RRNS-map codes. This is because $d^{RRNS}(\mathbf{a})$ and $L^{RRNS}(\mathbf{a})$ are related to the radius and the length along the axis of the cylinder, respectively, and the volume of the cylinder is bounded from above by the volume of the unit hyper cube. Thus, we have

$$d^{RRNS}(\mathbf{a}) = O(L^{RRNS}(\mathbf{a})^{-\frac{1}{N-1}}). \tag{4.41}$$

Figure 4.4 shows a concrete example of this trade-off. For each $\mathbf{a}$ in $\mathcal{A}_{50}$ with $N = 5$ and $W = [0, 1]$ (no side information), the minimum distance of the RRNS-map code $d^{RRNS}(\mathbf{a})$ is calculated and plotted as a function of the stretch factor $L^{RRNS}(\mathbf{a})$ in log-scale (black and gray dots). The dashed line provides the reference scaling, $\log d = -\frac{1}{N-1} \log L$, according to the prediction in (4.41). Black and gray dots shows to the RRNS-map codes, where black dots highlight candidate codes with large minimum distances that are greater than 95% of the prediction. There are 1156 of such RRNS-map codes with large minimum distances among 15279 $\mathbf{a}$'s in $\mathcal{A}_{50}$. Those RRNS codes fill in the gaps between the shift-map codes with different α's (red diamonds). In

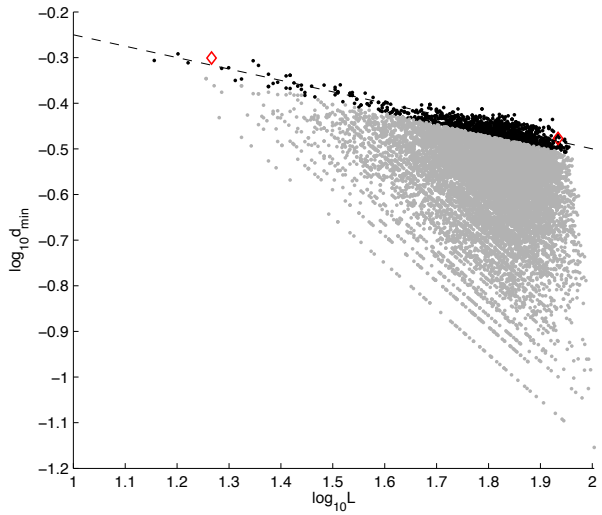

Figure 4.4: The trade-off between stretch factor and minimum distance for RRNS-map codes (black and gray dots), compared with shift-map codes (red diamonds) for $\mathcal{A}_{50}$. The dashed line represents predicted scaling with slope $-\frac{1}{N-1}$ between stretch factor and minimum distance in (4.41). Black dots highlight RRNS-map codes with large minimum distances that is close to the predicted scaling.

this sense, we say that the RRNS-map code generalizes the shift-map code.

In sum, good RRNS-map codes without side information are found, which are used as candidates for good RRNS-map codes with side information. In the following section, we search for codes among these RRNS-map codes found here that satisfy an additional requirement; the minimum distance scales well as more side information is revealed to the decoder

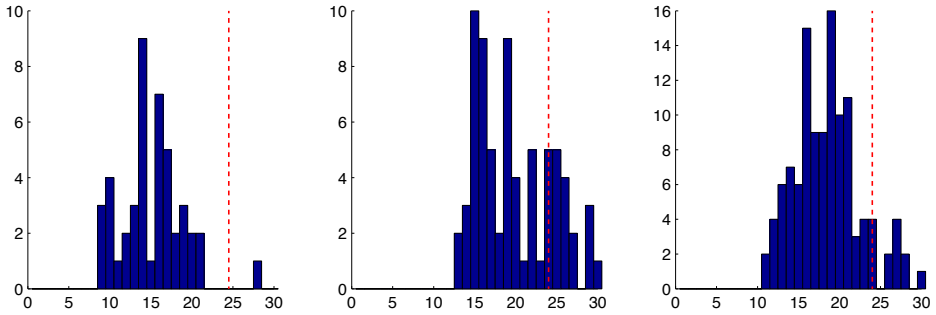

Figure 4.5: Histograms of number of neighbors of RRNS-map codes with $\mathbf{a}_1 = (5, 7, 9, 11, 17)$ (left), $\mathbf{a}_2 = (5, 7, 13, 23, 27)$ (middle), and $\mathbf{a}_3 = (5, 11, 19, 37, 47)$ (right) in the projected codebook $\mathcal{X}^*$. Red dashed lines show the densest possible number of neighbors by a regular lattice in the same in a $(N-1) = 4$ dimensional space.

4.4.4 Cylinder packing versus sphere packing

To further understand the cylinder packing problem, we compare the local structures of the RRNS-map code and of lattice. Finding the lattice with the largest density in a given dimension is equivalent to finding a way to pack as many as spheres with the same radii so that they barely touch with one another. The maximum number of spheres touching one at the center is called the kissing number and finding the largest possible kissing number in a given dimension is *the kissing number problem* [29]. The centers of spheres comprise a lattice codebook and used to encode discrete source.

The cylinder packing and the sphere packing are compared in the $N-1$ dimensional orthogonal hyperplane H. Note that the former controls the direction of the cylinders in N dimensional torus such that their projections are tightly packed in H while the latter concerns with the tightest sphere

packing directly in H. Thus, in the $(N-1)$-dimensional space, the sphere packing puts an upper bound on the densest cylinder packing.

We report that good RRNS-map codes have well-spaced codewords and close to lattice codes. Among the good RRNS codes with $N = 5$ found in the previous subsection (black dots in Figure 4.4), three parameters that satisfies additional constraints in the next section are: $\mathbf{a}_1 = (5, 7, 9, 11, 17)$, $\mathbf{a}_2 = (5, 7, 13, 23, 27)$, and $\mathbf{a}_3 = (5, 11, 19, 37, 47)$. For each parameter, Voronoi regions of $\mathcal{X}^*$ and corresponding numbers of neighbors are calculated by the quickhull algorithm [7] ported into Matlab [62]. The Voronoi region of a point $\mathbf{x}_c^* \in \mathcal{X}^*$ is defined as follows:

$$\begin{aligned} V_{\mathbf{a}}(\mathbf{x}_c^*) &= \{\mathbf{x} \in \mathcal{X}^* \,|\, |\mathbf{x} - \mathbf{x}_c^*| < |\mathbf{x} - \mathbf{x}'|, \\ &\qquad \mathbf{x}' \in \mathcal{X}^*, \mathbf{x}' \neq \mathbf{x}_c^*\}\,, \end{aligned} \tag{4.42}$$

where $|\cdot|$ is the Euclidean distance. The histograms of numbers of neighbors for these RRNS-map codes are shown in Figure 4.5. In the four dimensional space, the largest attainable kissing humber is 24 with lattice D_4 [29], shown in red dashed lines in Figure 4.5. The chosen RRNS-map codes have the number of neighbors (15.2 ± 3.8, and 19.8 ± 4.6, and 18.5 ± 4) similar to the kissing number of D_4. This hints that good RRNS-map codes are indeed well-spaced in the $(N-1)$ dimensional orthogonal hyperplane close to lattice codes.

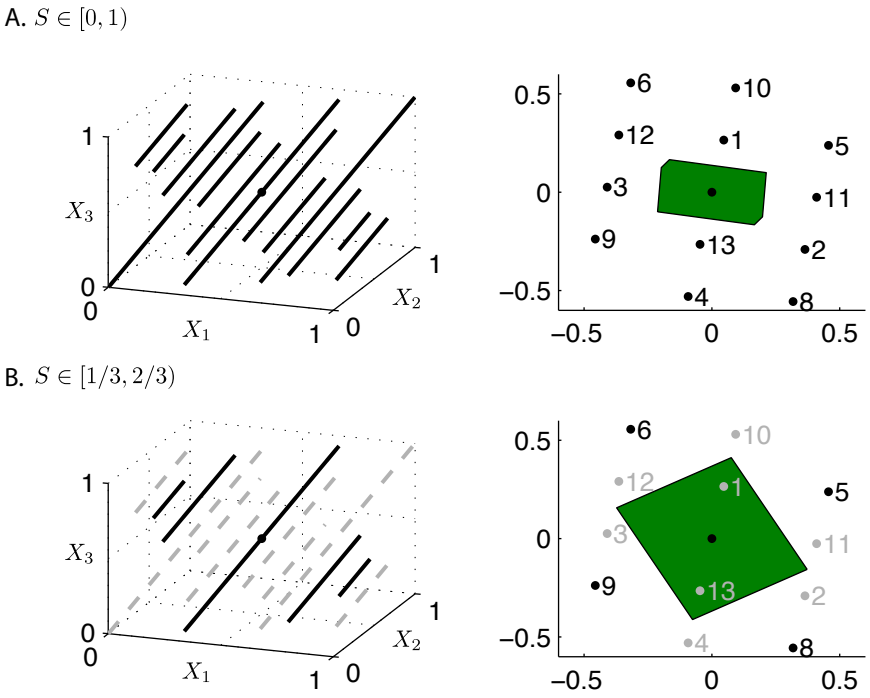

Figure 4.6: The RRNS-map codebook $\mathcal{X}^{RRNS}$ (left) and its projection to the orthogonal plane $\mathcal{X}^*$ (right) either without (A) and with (B) side information at decoder. Numbers in the right denote the order as the source increases from 0 and the green region is the Voronoi region of the center point.

4.5 Dynamic decoding of RRNS-map codes with side information

4.5.1 The RRNS-map codebook shrinks with side information at the decoder.

Now, let us design the RRNS-map codebook so that side information at the decoder is efficiently combined and results in a lower distortion. With side information W, the decoder only need to use the partial codebook $\mathcal{X}^{RRNS}(W)$. The size of this partial codebook decreases as more side information is revealed

to the receiver (equivalently, as $|W|$ decreases). The property we require for good RRNS-map codes is that this partial codebook is maximally separated in the codeword space and the minimum distance of $\mathcal{X}^{RRNS}(W)$ increases as $|W|$ decreases.

For example, Figure 4.6 illustrates the RRNS-map code with $\mathbf{a} = (3, 5, 7)$. In Figure 4.6A there is no additional information about S. Thus, the decoder must search for entire codebook $\mathcal{X}^{RRNS}$ in the left panel. Projections of the codebook, $\mathcal{X}^*([0, 1])$ is shown in the right panel with green area representing the Voronoi region of the center point. In Figure 4.6B, side information $S \in W = [1/3, 2/3]$ is provided to the receiver. Thus, the receiver needs to consider only a subset of the codebook, $\mathcal{X}^{RRNS}([1/3, 2/3])$, which contains five segments shown in black in the left panel. The number of segments needed for decoding decreases from 12 to 5. The right panel shows the projection of those five active segments in black while gray points do not need to be considered. Because the partial codebook is well separated, the side information results in increased distances between neighbors. Consequently, the Voronoi region (in green) of the center point increases.

Next, this shrinkage of the code book is quantified as a function $|W|$. Specifically, the cardinality of the the discrete codebook, $|\mathcal{X}^*(W)|$, linearly scales with $|W|$, as summarized in the following theorem.

Theorem 2. *Given a RRNS-map code with* $\mathbf{a}$ *satisfying (4.19), the cardinality of* $\mathcal{X}^*(W)$*, the intersections between the partial codebook* $\mathcal{X}^{RRNS}(W)$ *and the*

orthogonal hyperplane H in (4.25), is:

$$1+\sum_{i=1}^{N}\left(\lfloor a_n|W|\rfloor-1\right)\leq|\mathcal{X}^*(|W|)|\leq 1+\sum_{i=1}^{N}\left(\lfloor a_n|W|\rfloor\right) \tag{4.43}$$

Proof. The theorem follows from the same argument as in Lemma 5 with a slight modification that the support of the source considered are reduced by a factor of $|W|$. $\mathcal{X}^{RRNS}(|W|)$ intersects with $H_n(\mathbf{a})$ when corresponding source is $S=\frac{i}{a_n}\in W$ for some integer i. Thus, there are $\lfloor a_n|W|\rfloor$ integers for i. If $i=0$ is included, it should not be included to avoid multiple count. Summing up for all a_n's and subtracting redundant counts for $S=0$, we have (4.43). □

For large N, one can ignore the constant in (4.43) and have the following corollary.

Corollary 3. *The number of active points scales with the length of the side information $|W|$.*

$$|\mathcal{X}^*(W)|\approx|\mathcal{X}^*||W|. \tag{4.44}$$

4.5.2 The minimum distance of the RRNS-map codebook increases with side information at the decoder.

With side information at the decoder, the minimum distance of the RRNS-map code increases. For this purpose, we add a constraint that requires the shrinking codebook $\mathcal{X}^*(W)$ has large minimum distances for a range of $|W|$. Starting from the candidates found in the previous subsection without side information, we search for RRNS-map codes satisfying this additional constraint.

Next, an upper bound of the minimum distance $\tilde{d}$ is derived as a function of $|W|$. The radius of cylinder packing is bounded from above by that of optimal sphere packing. Thus, the volume of a $(N-1)$-dimensional ball with radius $\tilde{d}/2$ multiplied by the number of points in $\mathcal{X}^*(W)$ is bounded from above by the the volume of H, which is a constant:

$$|\mathcal{X}^*(W)|\frac{\pi^{\frac{N-1}{2}}}{\left(\frac{N-1}{2}\right)!}\left(\frac{\tilde{d}}{2}\right)^{(N-1)} \leq \mathrm{Vol}(H), \tag{4.45}$$

where the formula for $(N-1)$-dimensional ball is used when N is an odd and $\mathrm{Vol}(H)$ represents the volume of H. Using the approximation in (4.44) for the number of balls and applying Stirling's formula $\left(\frac{N-1}{2}\right)! \approx \left(\frac{N-1}{2}\right)^{\left(\frac{N-1}{2}\right)} e^{-\left(\frac{N-1}{2}\right)}$, we have

$$\tilde{d} \leq 2\sqrt{\frac{N-1}{2\pi e}}\left(\frac{\mathrm{Vol}(H)}{|\mathcal{X}^*||W|}\right)^{1/(N-1)}. \tag{4.46}$$

Thus, for a fixed $\mathbf{a}$, the exponent of $\tilde{d}$ scales with $|W|$ as follows:

$$\log \tilde{d} = O\left(\frac{1}{N-1}\log\frac{1}{|W|}\right). \tag{4.47}$$

4.5.3 Examples of good RRNS-map codes with $N = 5$

Next, we report that there exists a family of RRNS-map code with the desired properties; number of active points scaling linearly with $|W|$ as predicted in (4.44) and minimum distance close to the upper bound in (4.47). L^{RRNS} and d_{min}^{RRNS} and are calculated for each $\mathbf{a} \in \mathcal{A}_{50}$ with $N = 5, a_{max} = 50$, and varying W. For each $\mathbf{a} \in \mathcal{A}_{50}$, W is varied such that its boundaries are aligned to points among $\{0.1, 0.2, \ldots 1\}$. The number of points in $\mathcal{X}^*(W)$ and

the minimum distance are averaged over intervals with the same length and, therefore, are functions of $|W|$.

Among $\mathcal{A}_{50}$, $\mathbf{a}_1 = (5, 7, 9, 11, 17)$, $\mathbf{a}_2 = (5, 7, 13, 23, 27)$, and $\mathbf{a}_3 = (5, 11, 19, 37, 47)$ produce the three largest minimum distances in Figure 4.7. The stretch factors of those codes are between those of shift-map codes with $\alpha = 2$ and 3 (shown in red diamonds). The left panel of Figure 4.8 shows the size of the shrinking codebook $\mathcal{X}^*(W)$ as a function of $|W|$. Circles in the figure are numerical calculations and dashed lines are analytical predictions from (4.44). The right panel of Figure 4.8 shows the minimum distance as a function of $|W|$. Circles in the figure are numerical results and dashed lines are least-mean-square fits of the numerical results. The slopes of the fits are -0.22, -0.23, and -0.22, which are close to the analytical prediction $-\frac{1}{N-1} = -0.25$ from (4.47). Thus, the size of the shrinking codebook and resulting minimum distance match the analytical predictions.

4.5.4 Probability of threshold error and distortion of RRNS-map codes decrease with side information at the decoder.

The threshold error probability of RRNS-map codes decreases as more side information is revealed to the decoder. For the RRNS-map codes with parameters $\mathbf{a}_1 = (5, 7, 9, 11, 17)$, $\mathbf{a}_2 = (5, 7, 13, 23, 27)$, and $\mathbf{a}_3 = (5, 11, 19, 37, 47)$, upper bounds of the probability of threshold errors (P_{th}) are calculated using the minimum distances and numbers of neighbors (See Appendix C.1). In Figure 4.9A, the probabilities of threshold errors for the RRNS-map codes

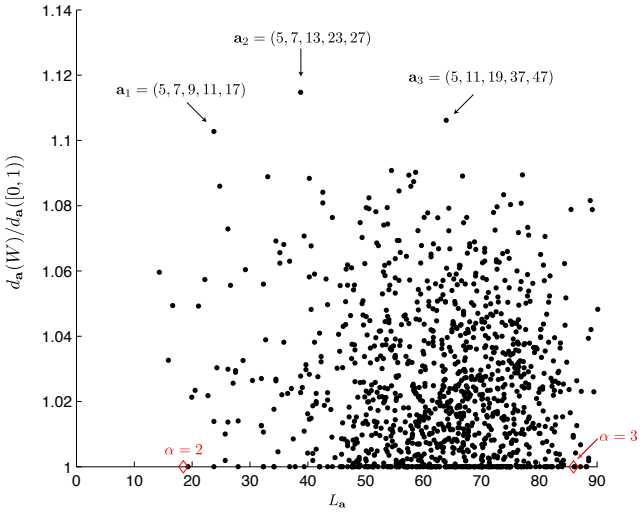

Figure 4.7: The minimum distance of the RRNS-map code increases with additional side information. For the good RRNS-map code identified in Figure 4.4, the amount of increase in the minimum distance compared to the minimum distance of the whole codebook ($d_{\mathbf{a}}(W)/d_{\mathbf{a}}([0,1))$) is plotted as a function of stretch factor. Three parameters for the RRNS-map with three largest increases in minimum distance are $\mathbf{a}_1 = (5, 7, 9, 11, 17), \mathbf{a}_2 = (5, 7, 13, 23, 27)$, and $\mathbf{a}_3 = (5, 11, 19, 37, 47)$. For comparison, shift-map codes with $\alpha = 2, 3$ are shown in red diamonds.

(triangles) and the shift-map codes (red dashed lines) are compared when $\sigma = 0.05$. As $|W|$ decreases (more side information), upper bounds of P_{th} of the RRNS-map codes decrease while lower bounds of P_{th} stay constant for the shift-map codes.

Consequently, the distortion of the RRNS-map decreases with more side information at the decoder. In Figure 4.9 right, the distortions of the RRNS-map codes are compared to shift-map, tent-map, and repetition codes. Upper bounds on the distortions of the RRNS-map codes (triangles) decreases

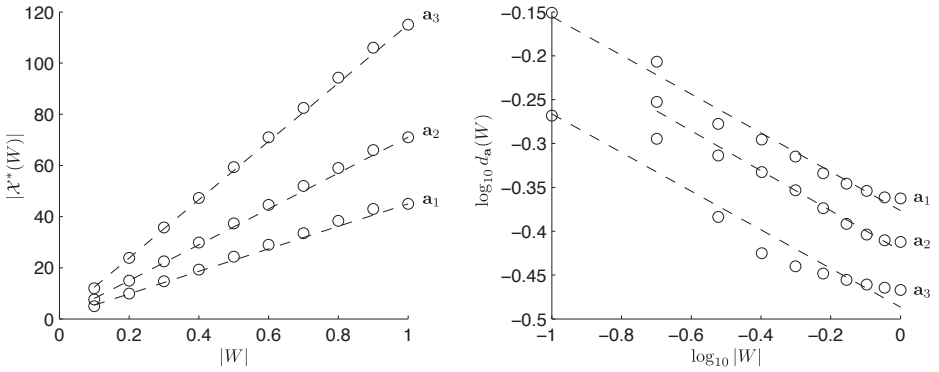

Figure 4.8: Numbers of active points (right) and minimum distances (right) and of good RRNS-map codes with $\mathbf{a}_1 = (5, 7, 9, 11, 17)$, $\mathbf{a}_2 = (5, 7, 13, 23, 27)$, and $\mathbf{a}_3 = (5, 11, 19, 37, 47)$ are plotted as as function of $|W|$. Circles indicate numerically found values, which agrees well with analytical predictions (4.44) and (4.47) shown in dashed lines.

as $|W|$ decreases. However, lower bounds on the distortions of the shift-map (red dashed lines) stay constant, so does the average distortion of the repetition code (green): $\frac{\sigma^2}{N}$. Those lower bounds are derived similarly to [92] (Appendix App:SM:LB). The black cross represents the distortion of the tent-map code [25, Eq. (16a) and (16b)]. Without the side information ($|W| = 1$), the upper bounds of the distortion of the RRNS-map codes with $\mathbf{a}_1 = (5, 7, 9, 11, 17)$, $\mathbf{a}_2 = (5, 7, 13, 23, 27)$ (from bottom to up) are similar to the lower bounds of the shift-map codes with $\alpha = 2$ and 3, respectively. As $|W|$ becomes smaller, the distortions of the RRNS-map codes decreases while those of the shift-map codes stays constant. Interestingly, the RRNS-map code with $\mathbf{a}_3 = (5, 11, 19, 37, 47)$ has larger distortion when $|W| = 1$, but the distortion becomes smaller than those RRNS-map codes with $\mathbf{a}_1 = (5, 7, 9, 11, 17)$,

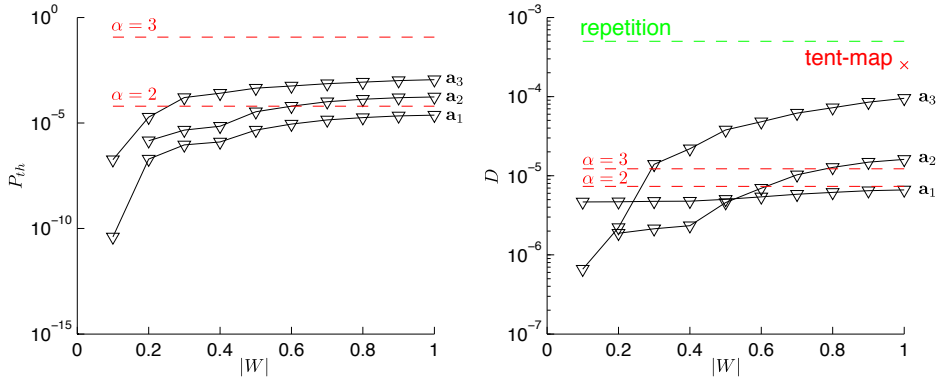

Figure 4.9: Probability of threshold error for the RRNS-map code decreases as the decoder has access to more side information (decreasing $|W|$) (left). The distortion of RRNS-map is compares to those of other analog codes (right). The distortions of RRNS-map codes without any side information ($|W| = 1$) are similar to those of shift-map codes with similar stretch factors ($\alpha = 2, 3$). But, as more side information is revealed to the decoder (as $|W|$ decreases), the distortion of the RRNS-code decreases without any change in encoding scheme. This contrasts the other analog codes whose distortions remain constant regardless of side information.

$\mathbf{a}_2 = (5, 7, 13, 23, 27)$ when $|W|$ approaches 0.

Different amount of decreases in distortions suggest that prior knowledge about the distribution of $|W|$ could results in a lower average distortion. For instance, if $|W|$ is uniformly distributed $\in [0, 1]$, the RRNS-map code with $\mathbf{a}_1$ and $\mathbf{a}_2$ produce lower average distortions than that with $\mathbf{a}_3$. On the other hand, if $|W|$ is concentrated to a smaller value and becomes larger occasionally, the RRNS-map code with $\mathbf{a}_3$ achieves a lower average distortion than those with $\mathbf{a}_1$ and $\mathbf{a}_2$.

4.6 Conclusions

We generalize the shift-map code to a new family called RRNS-map code so that the information from the sender and side information is effectively combined to reduce distortion of the source. The key idea is to introduce interleaving codebook by using relatively prime integers for generating codewords. Without side information at the decoder, the RRNS-map code produces the same distortion as the shift-map code with similar stretch factor does. With side information at the decoder, the RRNS-map code outperforms the shift-map code.

The dynamic scaling of the partial codebook of the RRNS-map code can be understand as an extension of binning schemes used to design discrete codebook for multi-user channels. Random binning is a widely used technique to allow multiple senders to provide complementary information about the source without cooperation [83, 100, 101]. We extend this idea to design a continuous codebook whose subset is well separated.

Finally, designing the RRNS-map code is an integer programming over relatively prime integers, which has a geometrical interpretation of cylinder packing. We numerically solve this problem to find example RRNS-map codes that have well-spaced partial codebooks. These codes result in excellent performance when varying degrees of side information are made available to the decoder.

Chapter 5

Conclusion

This work explores error-correcting codes in the contexts both of neuroscience and engineering. Specific forms of noises in the brain and communication systems are different. While the noise in a neuron's response is described by an inhomogeneous Poisson process, the noise in communication system generally matches additive Gaussian noises. Albeit such different characterizations, a general principle is found; redundancy reduces the effect of noise.

First, we study the mutual information between grid cells (GCs) in the enthorhinal cortex and self-location in a low-dimensional space as a function of tuning curve width. Multiple GC populations encodes the phase of the location using different scales. With the maximum likelihood (ML) decoding, narrower tuning increases the mutual information. With the neural network (NN) decoding, finite tuning is optimal. This dependence of information on tuning curve width is in contrast to classical population codes where a narrower tuning increases Fisher information regardless decoding method. The optimal tuning with the plausible NN decoding may explain relatively wide tuning actually observed in the brain.

Next, we further investigate on decoding multi-scale codes and propose

a decoding algorithm using belief propagation (BP). The gain of multi-scale codes in the higher tolerance to errors comes at a price of increased decoding complexity. Since the codewords of multi-scale codes interleave the entire codeword space, the decoder must search for a wide range to produce an estimate. Such a combinatorial problem is formulated as a subset selection problem on a graph with many cycles. We derive a belief propagation algorithm using the max-sum algorithm. Even though the underlying graph has many cycles, analysis and numerical simulations show the excellent convergence and accuracy when signal-to-noise ratio is high.

Furthermore, we show that the multi-scale codes studied in the context of neuroscience are useful for communication over a network. Using such codes, we propose a joint source-channel coding scheme for distributed coding where separate senders wish to transmit complimentary information without cooperation. We show that, with appropriate chose of moduli, the minimum distance of multi-scale code increases with side information. Thus, the receiver is able to decode the source with a lower threshold error and, consequently, a lower distortion. This scheme can be understood as an extension of binning schemes widely used in designing discrete codebooks.

For future works, we propose to further investigate on decoding of neural codes which, we believe, would shed light on new understandings in neuroscience. Previous studies on neural codes are mainly focused on encoding and the maximum amount of information from noisy neural responses under the assumption of ideal decoders. However, it remains obscure how the information

from one group of neurons is retrieved and used by another group of realistic neurons in the downstream. Along this direction, the proposed decoding based on BP offers a potential framework for efficient decoding with realistic neuron models and new understanding on how information is processed in the brain.

Appendices

Appendix A

Supplementary information for Chapter 2

A.1 Detailed Methods

A.1.1 Derivation of the maximum likelihood (ML) decoder

The ML decode's estimate of the encoded variable is the value that is maximally likely to give rise to the observed spike counts:

$$\hat{x}_{ML} = \arg\max_{\vec{x}} \log P(K|\vec{x}) \tag{A.1}$$

The maximum of the log of the likelihood equals the maximum of the likelihood because log is monotonic function. Under the assumption of independence in the spike counts of individual neurons after conditioning on their tuning curves, the likelihood $P(K|\vec{x})$ is the product of likelihoods of individual spike counts:

$$P(K|\vec{x}) = \prod_{n=1}^{N} \prod_{i=1}^{M} P(K_{ni}|\vec{x}), \tag{A.2}$$

where K_{ni} is the spike count of neuron i in network n ($n = 1, 2, \ldots, N, i = 1, 2, \ldots M$). Since the spatial location $\vec{x}$ produces a noisy phase $\vec{\phi}$ which, in turn, yields spike counts K_{ni}, in network n,

$$P(K|\vec{x}) = \prod_{n=1}^{N} \int_{\vec{\phi} \in [0,1)^D} \prod_{i=1}^{M} P(K_{ni}|\vec{\phi}) P(\vec{\phi}|\vec{\phi}_n(\vec{x})) d\vec{\phi}. \tag{A.3}$$

The first and the second terms in the integrand correspond to the Poisson variability in spikes and the sensing error in phase, respectively. Thus, from (2.1), (2.2), and (2.5), the likelihood has the following form:

$$\begin{aligned} P(K|\vec{x}) &= \prod_{n=1}^{N} \int_{\vec{\phi}\in[0,1)^D} \frac{e^{\sum_{i=1}^{M}\left\{K_{ni}\log\left(f_{ni}(\vec{\phi})dt\right)-f_{ni}(\vec{\phi})dt\right\}}}{\prod_{i=1}^{M} K_{ni}!} \frac{e^{\kappa_{sen}\cos 2\pi\left(\vec{\phi}-\frac{\vec{x}}{\lambda_n}\right)}}{I_o(\kappa_{sen})} d\vec{\phi} \\ &\propto \prod_{n=1}^{N} \int_{\vec{\phi}\in[0,1)^D} e^{\kappa_e \sum_{i=1}^{M}\left(K_{ni}\cos 2\pi\left(\vec{\phi}-\vec{\phi}^*_{ni}\right)\right)} e^{\kappa_{sen}\cos 2\pi\left(\vec{\phi}-\frac{\vec{x}}{\lambda_n}\right)} d\vec{\phi}. \qquad \text{(A.4)} \\ &= \prod_{n=1}^{N} e^{\kappa_e \sum_{i=1}^{M} K_{ni}\cos 2\pi\left(\frac{\vec{x}}{\lambda_n}-\vec{\phi}^*_{ni}\right)} * e^{\kappa_{sen}\cos\frac{2\pi\vec{x}}{\lambda_n}}, \qquad \text{(A.5)} \end{aligned}$$

where $\kappa_e = 1/(2\pi\sigma_e)^2$, $\kappa_{sen} = 1/(2\pi\sigma^{sen})^2$, (A.4) follows from removing terms not dependent on ϕ and x, and approximating $\sum_{i=1}^{M} f_{ni}(\vec{\phi})$ to a constant. The $*$ stands for convolution. The two terms in (A.5) have the following interpretations. The exponent of the first term is the "population average" of spike counts while the second term corresponds to a broadening of the population average peaks by the sensing noise.

By substituting (A.5) into (A.1), we have the maximum likelihood estimate as follows:

$$\hat{x}_{ML} = \arg\max_{\vec{x}} \sum_{n=1}^{N} \log\left(e^{\sum_{i=1}^{M} K_{ni}\kappa_e \cos 2\pi\left(\frac{\vec{x}}{\lambda_n}-\vec{\phi}^*_{ni}\right)} * e^{\kappa_{sen}\cos\frac{2\pi\vec{x}}{\lambda_n}}\right). \qquad \text{(A.6)}$$

When $\kappa_e \ll \kappa_{sen}$ ($\sigma_e \gg \sigma^{sen}$), convolution by the second term in (A.5) makes little contribution and therefore

$$P(K|\vec{x}) \approx \prod_{n=1}^{N} e^{\kappa_e \sum_{i=1}^{M} K_{ni}\cos 2\pi\left(\frac{\vec{x}}{\lambda_n}-\vec{\phi}^*_{ni}\right)}. \qquad \text{(A.7)}$$

Substituting (A.7) into (A.1), we have the ML decoder for the special case of a wide tuning curve width:

$$\hat{x}_{ML} = \arg\max_{\vec{x}} \sum_{n=1}^{N}\sum_{i=1}^{M} K_{ni}\cos 2\pi\left(\frac{\vec{x}}{\lambda_n}-\vec{\phi}^*_{ni}\right) \text{ if } \sigma_e \gg \sigma^{sen}. \qquad \text{(A.8)}$$

In numerical simulations, (A.6) is used in the following way. First, the range of spatial location $[R_l/2\ R_l/2]^D$ is densely quantized into H^D bins of equal size. Next, the likelihood is calculated for each bin. Finally, the location with maximum likelihood is chosen as the ML estimate.

A.1.2 Derivation of the neural network (NN) decoder by an associative learning rule

The neural network (NN) decoder is a read-out network connected to the MPC networks, that produces an estimate of the spatial location $\hat{x}$ directly from the MPC spike counts [89]. Specifically, the neural network (NN) decoder consists of H output neurons per dimension with preferred locations $x_j, j = 1, 2, \ldots, H^D$, that uniformly cover the coding range $[-R_l/2\ R_l/2]^D$. Each output neuron is connected to neurons in all MPC networks with predetermined synaptic weights w_{ni}^j, which are set up in advance by the training algorithm described below (Equation (A.14)). Individual neurons in the decoder network are driven by the MPC networks as follows:

$$h_j = \sum_{n=1}^{N} \sum_{i=1}^{M} w_{ni}^j K_{ni}, \tag{A.9}$$

where $j = 1, 2, \ldots, H$ is the neuron index for output neurons in the NN decoder, w_{ni}^j is the synaptic weight between output neuron j and neuron i ($i = 1, 2, \ldots, M$) from MPC network n ($n = 1, 2, \ldots N$). K_{ni} is the spike count from this cell. The estimate by the NN decoder, denoted as $\hat{x}_{NN}$, is the preferred spatial location of the maximally driven output neuron:

$$\hat{x}_{NN} = x_{j^*}^* \tag{A.10}$$

$$j^* = \arg\max_j h_j, \tag{A.11}$$

x_j^* is the preferred location of neuron j.

The synaptic weights between neurons in the MPC and in the decoder are trained using an associative rule. Initially, all the weights w_{ni}^j are set to zero. During training, the MPC networks and the decoder neurons are driven by a spatial location input $\vec{x}$, which produces noise-free firing rates in in the MPC neurons, according to their tuning curves (under the assumption that sensing noise is relatively low due to abundant familiar visual cues during training and that spiking error is effectively removed by averaging for sufficiently long time). During training, the output neurons' activities are assumed to be unimodal:

$$h_j^{train}(\vec{x}) = f_h(\vec{x}, \vec{x}_j^*) \tag{A.12}$$

$$f_h(\vec{x}, \vec{x}_j^*) = r'_m \prod_{d=1}^{D} \exp\left(-\frac{\left(\vec{x}(d) - \vec{x}_j^*(d)\right)^2}{2\sigma_h^2} \right), \tag{A.13}$$

where f_h is a unimodal tuning curve of Gaussian shape with width σ_h and x_j^* is the preferred location of neuron j. Weight w_{ni}^j is incremented by the amount of co-activation:

$$\Delta w_{ni}^j \quad = \quad r_{ni}(\vec{x}) h_j^{train}(\vec{x}). \tag{A.14}$$

Summing these increments once as x runs over all values of spatial location in the coding range $[-R_l/2\ R_l/2]^D$ produces the readout weights. These weights may be viewed as templates of the appropriate GPC phases for a given output neuron's preferred location.

A.1.3 Simplified neural network (NN) decoders

Next, we provide a closed-from approximation of the trained synaptic weight for the NN decoder. This analysis offers an efficient way for numerical simulation and, more importantly, reveals the equivalence of the ML and NN decoder under certain conditions, which is discussed in the following section.

First, the analytical form of the connectivity between the GC network and NN decoder, w_{ni}^j, is derived. The final weight, resulting from the rule A.14 applied to all locations $\vec{x}$, is:

$$w_{ni}^j \quad = \quad \frac{1}{H} \sum_{\vec{x} \in [-R_l/2\ R_l/2]^D} r_{ni}(\vec{x}) h_j(\vec{x}) \tag{A.15}$$

Given these weights, we may now derive the MPC-driven activation of the output neurons, by substituting (A.15) into (A.9):

$$\begin{aligned}
h_j \quad &\propto \quad \sum_{n=1}^{N} \sum_{i=1}^{M} \left(\sum_{\vec{x} \in [-R_l/2\ R_l/2]^D} r_{ni}(\vec{x}) h_j(\vec{x}) \right) K_{ni} \\
&= \quad \sum_{n=1}^{N} \sum_{\vec{x} \in [-R_l/2\ R_l/2]^D} \left(\sum_{i=1}^{M} r_{ni}(\vec{x}) K_{ni} \right) h_j(\vec{x}) \\
&\propto \quad \sum_{n=1}^{N} \sum_{\vec{x} \in [-R_l/2\ R_l/2]^D} \sum_{i=1}^{M} K_{ni} e^{\kappa_e \cos\left(2\pi\left(\frac{\vec{x}}{\lambda_n} - \vec{\phi}_{ni}^*\right)\right)} \left(e^{-\frac{\left(\vec{x}-\vec{x}_j^*\right)^2}{2\sigma_h^2}} \right) \\
&= \quad \sum_{n=1}^{N} \sum_{i=1}^{M} \sum_{\vec{x} \in [x_j^* - \lambda_n/2,\ x_j^* + \lambda_n/2]^D} K_{ni} e^{\kappa_e \cos\left(2\pi\left(\frac{\vec{x}}{\lambda_n} - \vec{\phi}_{ni}^*\right)\right)} \left(e^{-\frac{\left(\vec{x}-\vec{x}_j^*\right)^2}{2\sigma_h^2}} \right)
\end{aligned} \tag{A.16}$$

where $\kappa_e = 1/(2\pi\sigma_e)^2$.

In what follows, we assume $\sigma_h \ll \underline{\lambda} = E[\lambda_n]$, to obtain: (A.16):

$$
\begin{aligned}
h_j &\propto \sum_{n=1}^{N}\sum_{i=1}^{M}\sum_{\vec{x}\in[x_j^*-\lambda_n/2,\ x_j^*+\lambda_n/2]^D} K_{ni} e^{\kappa_e \cos\left(2\pi\left(\frac{\vec{x}}{\lambda_n}-\vec{\phi}_{ni}^*\right)\right)} e^{\kappa_h \cos\left(2\pi\left(\frac{\vec{x}-\vec{x}_j^*}{\lambda_n}\right)\right)} \\
&\propto \sum_{n=1}^{N}\sum_{i=1}^{M}\sum_{\vec{x}\in[x_j^*-\lambda_n/2,\ x_j^*+\lambda_n/2]^D} K_{ni} e^{\left\{\kappa_e \cos 2\pi\vec{\phi}_{ni}^* + \kappa_h \cos 2\pi\frac{\vec{x}_j^*}{\lambda_n}\right\}\cos 2\pi\frac{\vec{x}}{\lambda_n} + \left\{\kappa_e \sin 2\pi\vec{\phi}_{ni}^* + \kappa_h \sin 2\pi\frac{\vec{x}_j^*}{\lambda_n}\right\}\sin 2\pi\frac{\vec{x}}{\lambda_n}} \\
&\propto \sum_{n=1}^{N}\sum_{i=1}^{M} K_{ni} I_o(\kappa_{dec}(\vec{x}_j^*)) \qquad \text{(A.17)}
\end{aligned}
$$

where the concentrations κ_h and κ_{dec} are defined as follows:

$$\kappa_h = \frac{\underline{\lambda}^2}{(2\pi\sigma_h)^2} \qquad \text{(A.18)}$$

$$\kappa_{dec}(\vec{x}) = \sqrt{\kappa_e^2 + \kappa_h^2 + 2\kappa_e\kappa_h \cos 2\pi\left(\frac{\vec{x}}{\lambda_n} - \vec{\phi}_{ni}^*\right)} \qquad \text{(A.19)}$$

Equation (A.17) follows from the fact that the Gaussian and circular normal are close when the variance is low (the concentration is high). (A.17) is because the summation of circular normal form is approximately equal to the integral over a period, which is equal to the zeroth order Bessel function with argument κ_{dec}.

Therefore, substituting (A.17) into (A.10), we have a simplified NN decoder:

$$\hat{x}_{NN} = \arg\max_{\vec{x}} \sum_{n=1}^{N}\sum_{i=1}^{M} K_{ni} I_o(\kappa_{dec}(\vec{x})), \qquad \text{(A.20)}$$

where $\kappa_{dec}(\vec{x})$ is defined in (A.19).

In numerical simulations, we use (A.17) to calculate activities of read-out neurons and maximum of which coincides with the NN decode in (A.20). The results of this simplification were identical to those by training weights according to the update rule (A.14) (data not shown).

A.1.4 When is the NN decoder close to the ML decoder?

Now, we compare the NN decoder with the ML decoder and identify conditions under which they are close. Depending on the relative size between

σ_e and σ^{sen} and the order of σ_h, we consider four cases. For each case, the following polynomial approximation of the Bessel function [3] is used:

$$I_o(z) \approx \begin{cases} 1 + 3.52\left(\frac{z}{3.75}\right)^2 & \text{if } z < 3.75 \\ \frac{e^z}{\sqrt{2\pi z}} & \text{otherwise .} \end{cases} \tag{A.21}$$

i) $\sigma_e \ll \sigma^{sen}$ **and** $\sigma_h \sim \frac{1}{H} \ll 1$**:** When both σ_e and σ_h are small, κ_{dec} is large. Thus, substituting the polynomial approximation of Bessel function (A.21) for large κ_{dec} to (A.20), we have

$$\hat{x}_{NN} \approx \arg\max_{\vec{x}} \sum_{n=1}^{N} \sum_{i=1}^{M} K_{ni} \frac{e^{\kappa_{dec}}}{\sqrt{2\pi\kappa_{dec}}} \tag{A.22}$$

ii) $\sigma_e \gg \sigma^{sen}$**,** $\sigma_e \gg \frac{1}{2\pi}$**, and** $\sigma_h \sim \frac{1}{H} \ll 1$**:** In this case, we have $\kappa_e \ll 1$ and $\kappa_e \ll \kappa_h$. Under this condition, κ_{dec} is close to k_h and we use this fact for a simplfication:

$$\kappa_{dec} = \sqrt{\kappa_e^2 + \kappa_h^2 + 2\kappa_e\kappa_h \cos 2\pi\left(\frac{\vec{x}}{\lambda_n} - \vec{\phi}_{ni}^*\right)} \tag{A.23}$$

$$\approx \kappa_h \sqrt{1 + 2\frac{\kappa_e}{\kappa_h} \cos 2\pi\left(\frac{\vec{x}}{\lambda_n} - \vec{\phi}_{ni}^*\right)} \tag{A.24}$$

$$\approx \kappa_h + \kappa_e \cos 2\pi\left(\frac{\vec{x}}{\lambda_n} - \vec{\phi}_{ni}^*\right). \tag{A.25}$$

Substituting (A.21) and (A.25) into (A.20), we have

$$\hat{x}_{NN} = \arg\max_{\vec{x}} \sum_{n=1}^{N}\sum_{i=1}^{M} K_{ni} \frac{e^{\kappa_{dec}}}{\sqrt{2\pi\kappa_{dec}}} \tag{A.26}$$

$$\approx \arg\max_{\vec{x}} \sum_{n=1}^{N}\sum_{i=1}^{M} K_{ni} \frac{e^{\kappa_h+\kappa_e\cos 2\pi\left(\frac{\vec{x}}{\lambda_n}-\vec{\phi}^*_{ni}\right)}}{\sqrt{\kappa_h+\kappa_e\cos 2\pi\left(\frac{\vec{x}}{\lambda_n}-\vec{\phi}^*_{ni}\right)}} \tag{A.27}$$

$$\approx \arg\max_{\vec{x}} \sum_{n=1}^{N}\sum_{i=1}^{M} K_{ni} e^{\kappa_e\cos 2\pi\left(\frac{\vec{x}}{\lambda_n}-\vec{\phi}^*_{ni}\right)} \tag{A.28}$$

$$\approx \arg\max_{\vec{x}} \sum_{n=1}^{N}\sum_{i=1}^{M} K_{ni} \left(1+\left(\kappa_e\cos 2\pi\left(\frac{\vec{x}}{\lambda_n}-\vec{\phi}^*_{ni}\right)\right)\right) \tag{A.29}$$

$$\approx \arg\max_{\vec{x}} \sum_{n=1}^{N}\sum_{i=1}^{M} K_{ni} \left(\cos 2\pi\left(\frac{\vec{x}}{\lambda_n}-\vec{\phi}^*_{ni}\right)\right) \tag{A.30}$$

This leads to the same expression as the ML decoder under the assumption of negligible sensing noise (A.8).

iii) $\sigma^{sen} > 0$**,** $\sigma_e \ll \sigma^{sen}$**, and** $\sigma_h \sim \underline{\lambda}$**:** This is the opposite case of ii: $\kappa_e \gg 1 \gg \kappa_h$ and κ_{dec} is approximated around k_e.

$$\kappa_{dec} = \sqrt{\kappa_e^2+\kappa_h^2+2\kappa_e\kappa_h\cos 2\pi\left(\frac{\vec{x}}{\lambda_n}-\vec{\phi}^*_{ni}\right)} \tag{A.31}$$

$$\approx \kappa_e\sqrt{1+2\frac{\kappa_h}{\kappa_e}\cos 2\pi\left(\frac{\vec{x}}{\lambda_n}-\vec{\phi}^*_{ni}\right)} \tag{A.32}$$

$$\approx \kappa_e+2\frac{\kappa_h}{\kappa_e}\cos 2\pi\left(\frac{\vec{x}}{\lambda_n}-\vec{\phi}^*_{ni}\right) \tag{A.33}$$

Substituting (A.21) and (A.33) to (A.20), we have

$$\hat{x}_{NN} = \arg\max_{\vec{x}} \sum_{n=1}^{N}\sum_{i=1}^{M} K_{ni} \frac{e^{\kappa_{dec}}}{\sqrt{2\pi\kappa_{dec}}} \tag{A.34}$$

$$\approx \arg\max_{\vec{x}} \sum_{n=1}^{N}\sum_{i=1}^{M} K_{ni} \frac{e^{\kappa_e + 2\frac{\kappa_h}{\kappa_e}\cos 2\pi\left(\frac{\vec{x}}{\lambda_n} - \vec{\phi}^*_{ni}\right)}}{\sqrt{2\pi\kappa_e + 2\frac{\kappa_h}{\kappa_e}\cos 2\pi\left(\frac{\vec{x}}{\lambda_n} - \vec{\phi}^*_{ni}\right)}} \tag{A.35}$$

$$\approx \arg\max_{\vec{x}} \sum_{n=1}^{N}\sum_{i=1}^{M} K_{ni} e^{2\frac{\kappa_h}{\kappa_e}\cos 2\pi\left(\frac{\vec{x}}{\lambda_n} - \vec{\phi}^*_{ni}\right)} \tag{A.36}$$

$$\approx \arg\max_{\vec{x}} \sum_{n=1}^{N}\sum_{i=1}^{M} K_{ni} \left(1 + \left(2\frac{\kappa_h}{\kappa_e}\cos 2\pi\left(\frac{\vec{x}}{\lambda_n} - \vec{\phi}^*_{ni}\right)\right)\right) \tag{A.37}$$

$$\approx \arg\max_{\vec{x}} \sum_{n=1}^{N}\sum_{i=1}^{M} K_{ni} \left(\cos 2\pi\left(\frac{\vec{x}}{\lambda_n} - \vec{\phi}^*_{ni}\right)\right) \tag{A.38}$$

This leads to the same expression as the ML decoder under the assumption of negligible sensing noise (A.8).

iv) $\sigma_e \gg \sigma^{sen}$, and $\sigma_h \sim \lambda$: When both σ_e and σ_h are large, κ_{dec} is small. Thus, substituting the polynomial approximation of Bessel function (A.21) for small κ_{dec} to (A.20), we have

$$\begin{aligned}
\hat{x}_{NN} &= \arg\max_{\vec{x}} \sum_{n=1}^{N}\sum_{i=1}^{M} K_{ni} \left(1 + \frac{3.52}{3.75^2}\left(\kappa_e^2 + \kappa_h^2 + 2\kappa_e\kappa_h \cos 2\pi\left(\frac{\vec{x}}{\lambda_n} - \vec{\phi}^*_{ni}\right)\right)\right) \\
&= \arg\max_{\vec{x}} \sum_{n=1}^{N}\sum_{i=1}^{M} K_{ni} \left(\frac{4 + \kappa_e^2 + \kappa_h^2}{2\kappa_e\kappa_h} + \cos 2\pi\left(\frac{\vec{x}}{\lambda_n} - \vec{\phi}^*_{ni}\right)\right) \\
&= \arg\max_{\vec{x}} \sum_{n=1}^{N}\sum_{i=1}^{M} K_{ni} \left(\cos 2\pi\left(\frac{\vec{x}}{\lambda_n} - \vec{\phi}^*_{ni}\right)\right)
\end{aligned} \tag{A.39}$$

This leads to the same expression as the ML decoder under the assumption of negligible sensing noise (A.8).

Thus, for cases ii, iii, and iv, the NN decoder is equivalent to an ML decoder that is ignorant of sensing noise.

Comparing with the numerical results, case i and ii correspond to the NN decoder with a narrow σ_h, Figure 2.5. From the latter, we know that the

NN decoder is close to the ML decoder ignorant of sensing noise. However, interestingly, the total MI of the NN decoder is very close to that of the ML decoder for a larger σ_e (Figure 2.5D) becuase the effect of sensing noise is not significant when $\sigma_e \gg \sigma^{sen}$.

On the other hand, case iii and iv correspond to the NN decoder with wider σ_h, Figure 2.6. In Figure 2.6D, the total MI of the NN decoder becomes closer to the ML decoder for both narrow and wide σ_e's. In other words, increasing σ_h rescues the catastrophic failure of the NN decoder with both narrow σ_h and σ_e. Still, there is a gap between the NN and the ML decoders, indicating the failure of including the effect of the sensing noise. This gap becomes negligible for wider $\sigma_e \gg \sigma^{sen}$.

A.1.5 Details of numerical simulations

In numerical simulations for the ML decoder, (A.6) is used in the following way. First, the range of spatial location $[-R_l/2 \;\; R_l/2]^D$ is densely quantized into H^D bins with equal size. Next, then likelihood is calculated for each bin. Finally, location with maximum likelihood is chosen as the ML estimate.

For the NN decoder, we use (A.17) to calculate activities of output neurons. The preferred locations are set to the same quantized bins as the ML decoder. The preferred location of the maximally responding output neurons coincides with the NN decode in (A.20). The results of this simplification were identical to those by training weights according to the update rule (A.14) (data not shown).

For numerical simulations for one-dimensional spatial location ($D = 1$), the coding range $[-150\ 150]$ ($R_l = 300$) is quantized into $H = 19200$ bins with equal size $\frac{1}{64}$. For the grid networks, $N = 8, \lambda_n = \{30, 34, 38, 42, 46, 50, 54, 58\}$, $M = 256$, and $r_m \Delta t = 1$. In Figure 3 for the ML decoder, sensing noise σ^{sen} and grid cell tuning curve width σ_e are varied independently. In Figure 4 for the NN decoder with narrow σ_h, the sensing noise σ^{sen} is fixed to 0.1 and σ_e is varied. Decoding tuning curve $\kappa_{dec} = \kappa_e = \frac{1}{(2\pi\sigma_e)^2}$ under the assumption that σ_h is in the order of $\frac{1}{H}$ and $\sigma_h \ll \underline{\lambda}\sigma_e$. In Figure 5 for the NN decoder with varying σ_h, the sensing noise σ^{sen} is fixed to 0.1 and σ_e and σ_h are varied independently.

A.2 The stretch factor increases with decreasing σ_e.

To provide the connection between the FI and geometric view of decoding, let us calculate the stretch factor of GC. The noiseless instantaneous firing rate vector of GC in network n is denoted as $\vec{r}_n(\phi)$ with each element having identical circular normal tuning curve translated by its preferred phase ϕ_{ni}:

$$\vec{r}_n(\phi) = [r_{n1}(\phi), r_{n2}(\phi), \ldots r_{nM}(\phi)] \tag{A.40}$$

$$r_{ni}(\phi) = r_m e^{\kappa_e\{\cos(2\pi(\phi_n-\phi_{ni}))-1\}}. \tag{A.41}$$

First, $\vec{r}_n(\phi)$ lies on the M-sphere with a radius R. This is readily seen by that Eucledian norm of any $\vec{r}_n(\phi)$ is a constant, regardless of ϕ:

$$|r_{ni}(\phi)|_2 = r_m^2 \sum_{i=1}^{M} e^{2\kappa_e\{\cos(2\pi(\phi_n-\phi_{ni}))-1\}} \tag{A.42}$$

$$= r_m^2 M \frac{I_o(2\kappa_e)}{e^{2\kappa_e}}, \tag{A.43}$$

where (A.43) follows from turning the summation into integral under assumption of a large M. Thus

$$R(\sigma_e) = \frac{r_m}{e^{\kappa_e}}\sqrt{MI_o(2\kappa_e)}. \tag{A.44}$$

Next, the stretch factor $L_n(\sigma_e)$ is calculated from the arc-length:

$$L_n(\sigma_e) = \int_0^1 \left|\frac{\partial \vec{r}_n}{\partial d\phi_n}\right| d\phi_n. \tag{A.45}$$

Since

$$\left|\frac{\partial \vec{r}_n}{\partial d\phi_n}\right| = r_m\sqrt{\sum_{i=1}^{M} e^{2\kappa_e\{\cos(2\pi(\phi_n-\phi_{ni}))-1\}}\sin^2(2\pi(\phi_n-\phi_{ni}))(2\pi\kappa_e)^2}$$

$$= \frac{\pi r_m}{e^{\kappa_e}}\sqrt{M2\kappa_e I_1(2\kappa_e)}, \tag{A.46}$$

$$L_n(\sigma_e) = \frac{\pi r_m}{e^{\kappa_e}}\sqrt{M2\kappa_e I_1(2\kappa_e)} \tag{A.47}$$

For $\sigma_e \ll 1$, $I_1(2\kappa_e) \approx I_o(2\kappa_e)$, therefore from (A.44) and (A.47)

$$\frac{L_n(\sigma_e)}{R(\sigma_e)} \approx \frac{1}{\sqrt{2}\sigma_e}. \tag{A.48}$$

This explains the denser packing of coding line with a narrow tuning curve widths and resulting higher threshold error probability.

A.3 Fisher information calculation

Fisher information (FI) provides an upper bound of total mutual information. This is because its estimate of mean squared error squared error is a lower bound, and since it does not account for the occurrence of nonlocal threshold errors. When threshold error does not occur, the posterior distribution of spatial location is unimodal and Gaussian [89]. Under this condition, with the assumption of a large number of neurons, the MI is related with the FI as follows [18]:

$$I(x;\hat{x}|\text{no threshold error}) = H(x) - \frac{1}{2}\log\frac{2\pi e}{|J(x)|}, \tag{A.49}$$

Thus, in general, the mutual information is bounded from above by (A.49):

$$I(x;\hat{x}) \leq I(x;\hat{x}|\text{no threshold error}) = \frac{1}{2}\log|J(x)| + \log\left(\frac{R_l^D}{\sqrt{2\pi e}}\right). \tag{A.50}$$

Next, the Fisher information about the spatial location $J(x)$ is calculated by summing FI from each network $J_n(x)$:

$$J(\vec{x}) \;=\; \sum_{n=1}^{N} J_n(\vec{x}) \tag{A.51}$$

The inverse of FI per network includes the effect of the sensing error and CR bound of reading out phase from Poisson spikes:

$$\frac{1}{J_n(\vec{x})} = \lambda_n^2\left(\sigma_{sen}^2 + \frac{1}{J_n(\vec{\phi}_n)}\right), \tag{A.52}$$

where $J_n(\vec{\phi}_n)$ is the FI due to Poission spikes:

$$\begin{aligned} J_n(\vec{\phi}_n) &= \frac{1}{D}\sum_{i=1}^{M}\left\langle -\frac{\partial^2 \log P(K_{ni}|\vec{\phi}_n)}{\partial^2 \vec{\phi}_n}\right\rangle_{K_{ni}} \\ &= \frac{r_m \Delta t}{(2\pi\sigma_e^2)^2}\sum_{i=1}^{M}\prod_{d'=1}^{D} e^{\left(\frac{\cos(2\pi(\phi_n^{(d')}-\phi_{ni}^{*(d')}))-1}{(2\pi\sigma_e)^2}\right)}\sin^2\left(2\pi\left(\phi_n^{(d)}-\phi_{ni}^{(d)*}\right)\right) \\ &= \left((2\pi)^2\frac{M r_m \Delta t}{D}\right)\kappa_e\left(\frac{I_1(\kappa_e)}{e^{\kappa_e}}\right)\left(\frac{I_0(\kappa_e)}{e^{\kappa_e}}\right)^{(D-1)} \end{aligned} \tag{A.53}$$

where superscript (d) represents the d-dimensional component, $\kappa_e = 1/(2\pi\sigma_e)^2$, I_1 and I_0 are the first and the zeroth order modified Bessel functions of the first kind, respectively.

Note that even if $J_n(\vec{\phi}_n)$ tends to infinity, $J_n(\vec{x})$ is finite due to the sensing noise (A.52). This explains the saturation of FI for narrow tuning curve width in the presence of sensing noise.

A.4 The posterior distribution and estimation of p_{th} for single-population codes.

The posterior distributions of single populations codes, from which FI and P_{th} in Figure 2.3F and G are calculated, are shown here.

Figure A.1 shows posterior probabilities of the ML decoder without sensing error $\sigma^{sen} = 0$. In the posterior, the width of the central peak w_c is given from the square root of the inverse $J(x)$ and then probability of threshold error is defined as $P(x - cw_c < \hat{x}_{ML} < x + cw_c|x)$, where $c = 6$ controls the confidence interval and is adjusted to exclude the main peak but include significant portions of threshold errors.

Neither switching the ML decoder to the NN decoder nor changing the tuning curve shape from circular normal to cosine shows qualitatively different posteriors, Figures A.2 and A.3.

When the sensing error with intermediate magnitude ($\sigma^{sen} = 0.05$) are added, the main peak of the posterior becomes wider and, consequently, the overall P_{th} decreases, compared to the case with $\sigma^{sen} = 0$, Figure A.4.

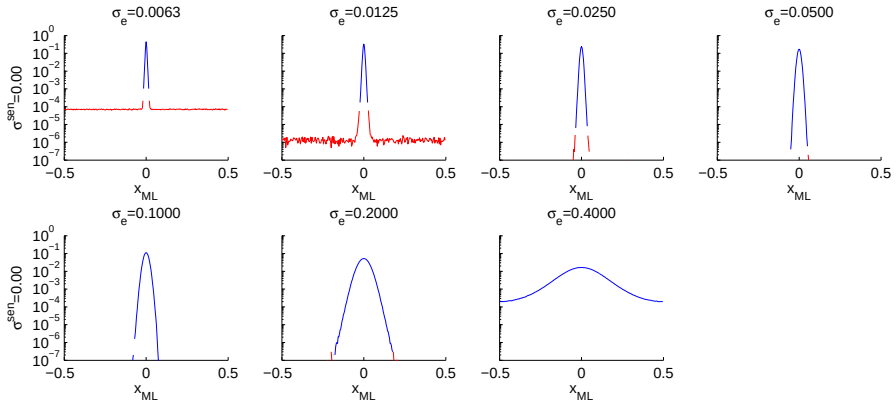

Figure A.1: Posterior probabilities $P(\hat{x}_{ML}|x = 0)$ for different tuning curve widths σ_e with the ML decoder and $\sigma^{sen} = 0$.

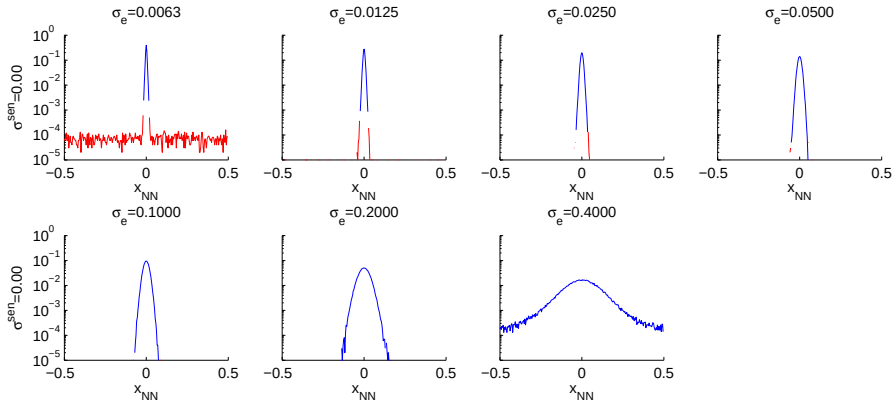

Figure A.2: Posterior probabilities $P(\hat{x}_{ML}|x = 0)$ with the NN decoder shows no significant difference.

However, the dependence of P_{th} on σ_e is preserved: $P_{th} > 0$ for a very narrow σ_e and $P_{th} = 0$ otherwise.

On the other hand, when the sensing noise becomes much larger $\sigma^{sen} = 0.2$, the posterior becomes wide, Figure A.5 and $P_{th} = 0$ for all σ_e in the simulated range.

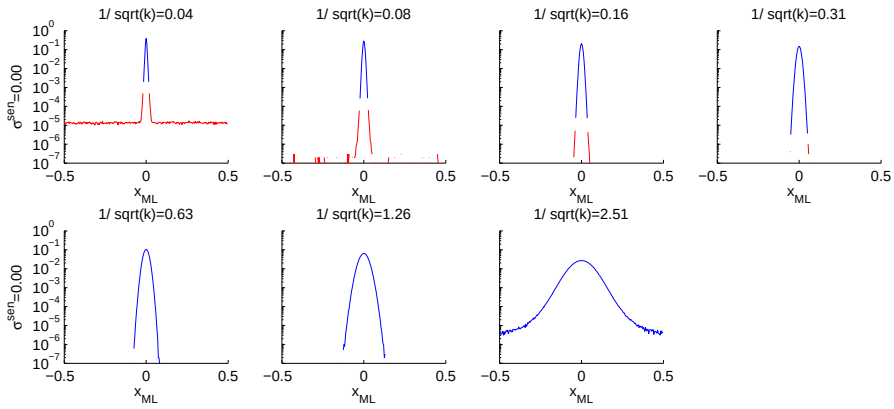

Figure A.3: Posterior probabilities $P(\hat{x}_{ML}|x = 0)$ with cosine tuning curve shows no significant difference.

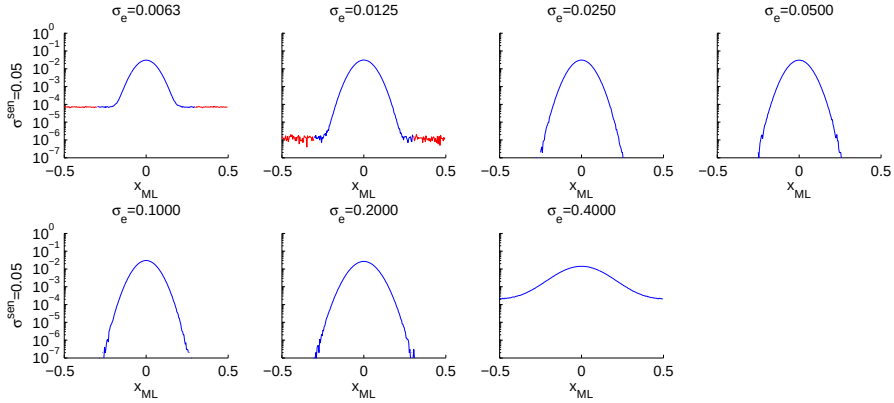

Figure A.4: Posterior probabilities $P(\hat{x}_{ML}|x = 0)$ with an intermediate sensing error ($\sigma^{sen} = 0.05$, the ML decoding, circular normal tuning curve)

A.5 The relative sizes of spike sample noise and sensing noise.

In numerical simulations, parameters are chosen such that the effects of sensing error and spike sample error are in the same order of magnitude. The sensing error is varied from 0 to 0.2. In each network with $D = 1$ and $M = 256$, as σ_e varied from 0.00625 to 0.4, $\sqrt{J(\phi)^{-1}}$ monotonically increases from 0.0031 to 0.096 (red dashed lines in Figure A.6).

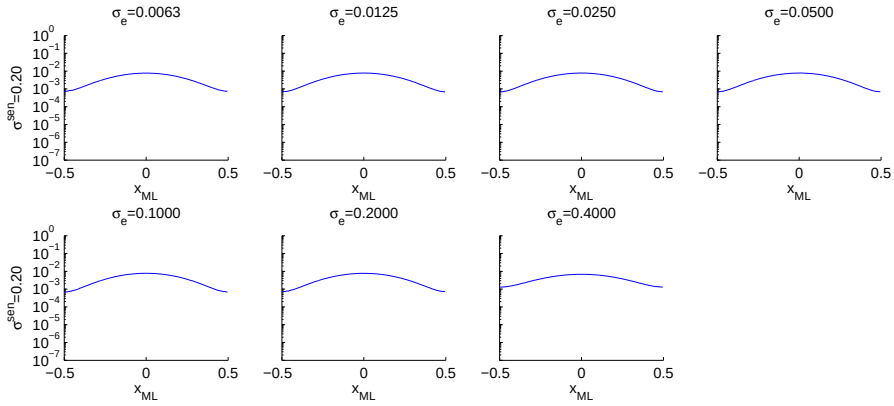

Figure A.5: Posterior probabilities $P(\hat{x}_{ML}|x = 0)$ with a large sensing error ($\sigma^{sen} = 0.2$, circular norma tuning curve, the ML decoding, circular normal tuning curve)

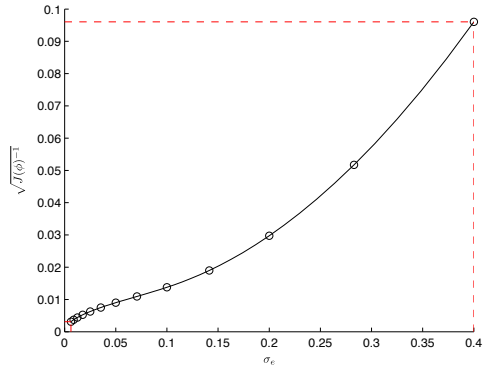

Figure A.6: $1/\sqrt{J(\phi)^{-1}}$ as a function of σ_e for $D = 1$ and $M = 256$

A.6 The posterior distributions and threshold errors of multi-scale population codes

Figures A.7 and A.8 show the posterior distributions by numerical simulations for multiple populations (with $N = 8$). The threshold error for multiple populations, defined as $P(|\hat{x}_{ML} - x| > \frac{1}{2}\min_n \lambda_n|x)$, is shown in red.

A too wide tuning curve $\sigma_e = 0.4$ results in large threshold error probabilities in both the ML and the NN decoder.

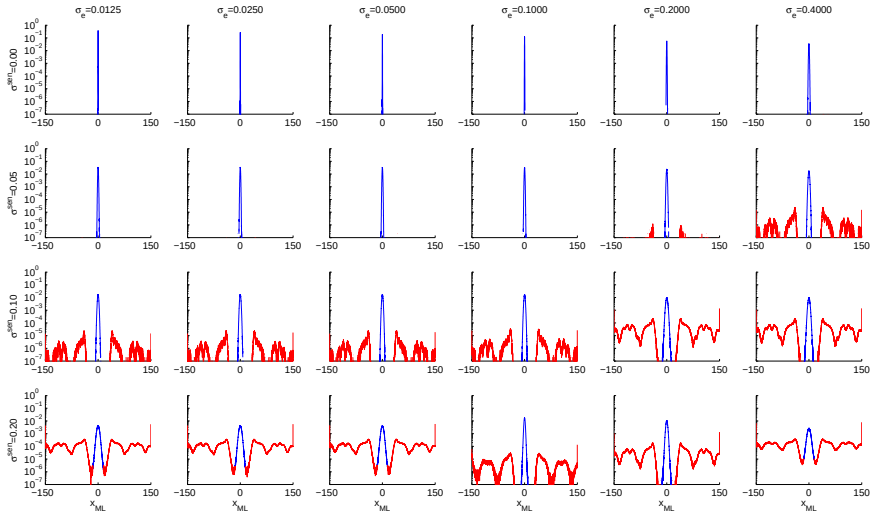

Figure A.7: $P(\hat{x}_{ML}|x=0)$, $N=8$, ML decoding. Red: threshold error.

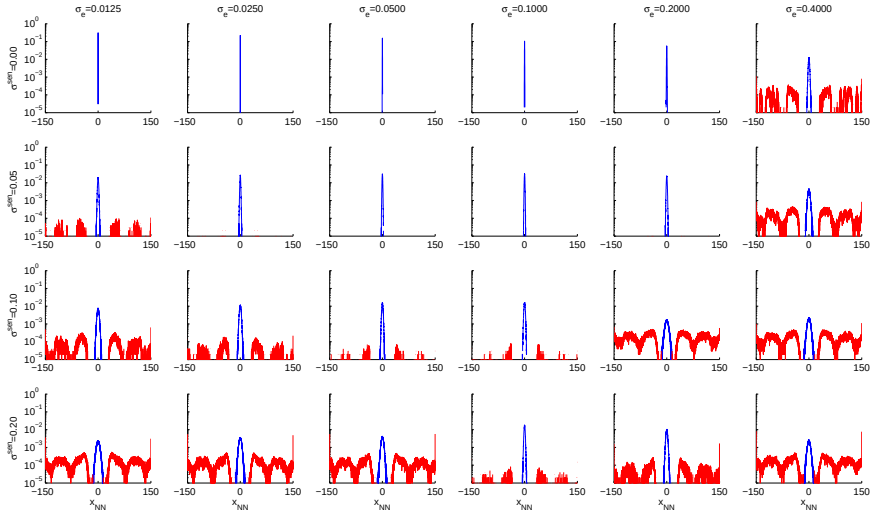

Figure A.8: $P(\hat{x}_{NN}|x=0)$, $N=8$, NN decoding. Red: threshold error.

A.7 Equivalence of noisy learning and increasing σ_h

When the NN decoder was derived, the sensing noise was assumed to be insignificant during training, due to the presence of rich spatial cues.

What happens when sensing noise is not negligible during training? Here, we show that training the NN decoder in the presence of sensing noise is roughly equivalent to increasing the during-training decoding tuning curve width σ_h, by an appropriately scaled version of the sensing error, $\bar{\lambda}\sigma_{sen}$. As we show in Figure 2.6, increasing the during-training encoder width by this amount brings the decoding performance of the NN decoder nearly to par with that of the ML decoder in the presence of sensing noise. Therefore, training decoder weights in the presence of sensing noise results in better decoder performance after training.

$$w_{ni}^{j'} = \frac{1}{H} \sum_{\vec{x}\in[-R_l/2,\, R_l/2]^D} r'_{ni}(\vec{x}) h_j(\vec{x}) \tag{A.54}$$

$$r'_{ni}(\vec{x}) = \int_{[0,1]^D} r_{ni}(\vec{\phi}_n) P(\vec{\phi}_n|\vec{x}) d_{\vec{\phi}_n}, \tag{A.55}$$

where the learning rule in (A.54) is defined similarly to (A.15) with noisy response in (A.55). When this noisy weight is used for decoding, the activity of decoding neuron follows in (A.16) becomes:

$$\begin{aligned} h'_j &\propto \sum_{n=1}^{N}\sum_{i=1}^{M} \sum_{\vec{x}\in[x_j-\lambda_n/2,\, x_j+\lambda_n/2]^D} K_{ni} \left(\int_{\vec{\xi}_n\in[0,1]^D} e^{\kappa_e \cos\left(2\pi\left(\frac{\vec{x}}{\lambda_n}-\vec{\phi}^*_{ni}-\vec{\epsilon}_n\right)\right)} e^{-\frac{1}{2}\left(\frac{\vec{\xi}_n}{\sigma^{sen}}\right)^2} d\vec{\xi} \right) e^{-\frac{\left(\vec{x}-\vec{x}^*_j\right)^2}{2\sigma_h^2}} \\ &\approx \sum_{n=1}^{N}\sum_{i=1}^{M} \sum_{\vec{x}'\in[x_j-\lambda_n/2),\, x_j+\lambda_n/2)]^D} K_{ni} e^{\kappa_e \cos\left(2\pi\left(\frac{\vec{x}'}{\lambda_n}-\vec{\phi}^*_{ni}\right)\right)} \left(e^{-\frac{\left(\vec{x}'-\vec{x}^*_j\right)^2}{2(\sigma_h^2+\lambda_n^2\sigma^2_{sen})}} \right), \end{aligned} \tag{A.56}$$

Note that (A.56) has the same form as noiseless activity in (A.16) with effective decoding tuning curve width σ'_h:

$$\sigma_h^{2\prime} = \sigma_h^2 + \lambda_n^2 \sigma_{sen}^2. \tag{A.57}$$

Thus, the sensing noise during the training is equivalent to the increase of σ_h at the NN decoder.

Sensing noise during the training effectively widens the bumps in the connectivity between GC and the NN decoder w_{ni}^j. The trained weight in (A.15) is convolved with sensing noise:

A.8 2D input shows qualitatively the same results as 1D

Figure A.9 shows numerical simulations for two-dimensional spatial location ($D = 2$). The number of neurons is reduced due to the memory and time requirements: $M = 32 \times 32, H = 300 \times 300$. The sensing noise $\sigma^{sen} = 0.1$ is fixed to 0.1 and σ_e is varied. σ_h is either narrow ($\kappa_{dec} = \kappa_e$) or wide $\sigma_h = 0.1\lambda$. Other parameters remain the same as for $D = 1$.

A.9 Hierachically nested spatial periods generate qualitatively the same results for optimal tuning curve width

Based on neurobiological findings for grid cells – the periods span the relatively small range of 30 cm to 3 m or about one decade in magnitude – we considered in this work a multiperiodic code with a small range of spatial periods. In other words, $\lambda_N \ll R_l$, and $\lambda_N/\lambda_1 \approx 2$ ($N = 8$). The periods λ_n were related through: $\lambda_n = 30 + 4(n-1)$. An alternative choice is to pick $\lambda_N = R_l$ ($N = 8$), with periods related according to $\lambda_n = \lambda_0 2^{n-1}$. Thus, $\lambda_0 = 3$ and $\lambda_N/\lambda_0 = 128$.

With a large hierarchical span of scales in the periods, sensing noise induces large threshold errors in the NN decoder, when σ_e is small (narrow tuning curves), Figure A.10; with ML decoding, narrower tuning curves are better. This result is in complete agreement with results when the periods are roughly similar in size and all much smaller than the range R_l. Thus, the code structure and the effect of tuning curve width on threshold error is not much different for multi-scale codes with a narrow versus large span of periods.

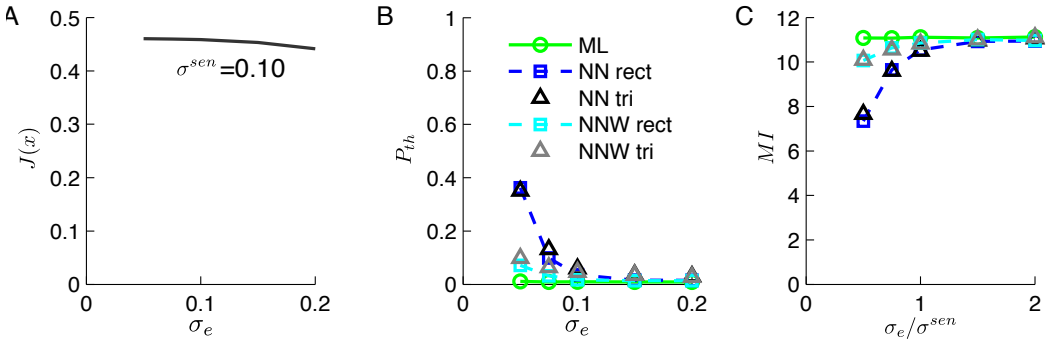

Figure A.9: Results for 2D multiperiod multipopulation coding. Results for two-dimensional spatial locations ($D = 2$, $\sigma^{sen} = 0.1$) shows the same pattern as is for the one-dimensional case. (A) Fisher information stays relatively constant in the presence of the sensing error. (B) With the ML decoder, the probability of threshold error (P_{th}) is close to zero (green circles). In contrast, the NN decoder with narrow $\sigma_h \sim \frac{1}{M} \ll 1$ results in a large P_{th} as tuning curve width becomes narrower. Here, blue square and black triangle represent rectangular and triangular lattices for arrangements of tuning curve in 2-dimensional spatial location The threshold error probability does not depend on the spatial arrangement. This large P_{th} is reduced by increasing $\sigma_h = 0.1\underline{\lambda}$, denoted as NNW in the legend. Again, the choice of the spatial arrangement makes no difference (cyan square and gray triangle for rectangular and triangular arrangements, respectively.) (C) Consequently, the total mutual information (MI) stays constant for the ML decoder but rapidly drops for the NN with narrower σ_h as the tuning curve width decreases. This drop in MI for NN decoder is rescued by increasing σ_h. There is no significant difference due to the choice of the lattice. Parameters used for numerical simulations are as follows: $R_l = 300$, $H = 300 \times 300$, $N = 8, \lambda_n = \{30, 34, 38, 42, 46, 50, 54, 58\}$, $\underline{\lambda} = 44$, $M = 32 \times 32$,, $r_m \Delta t = 1$, $\sigma^{sen} = 0.1$, $\sigma_h \sim \frac{1}{H}$ for the NN decoder and $\sigma_h = 0.1\underline{\lambda}$ for the NNW decoder. The number of samples for each condition is 10^4.

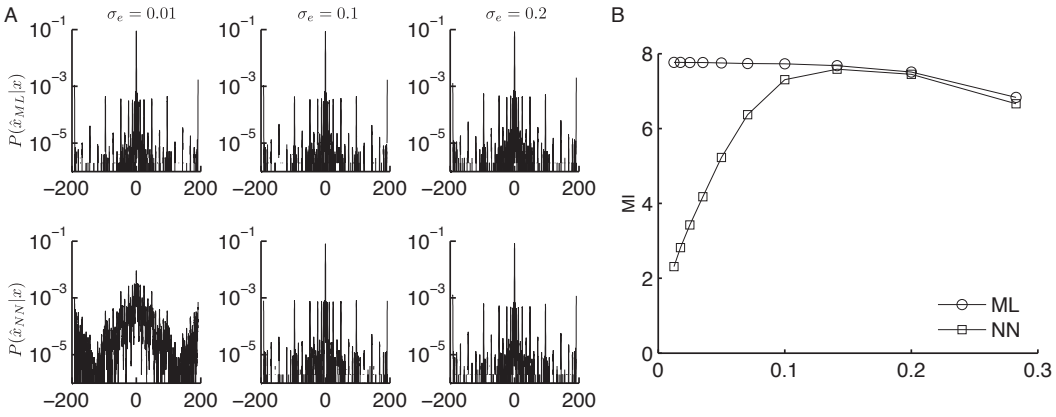

Figure A.10: A multi-scale code with hierarchically nested scales of periods exhibits the same dependence on tuning curve widths as for an MPC with distinct but similarly sized periods. (A) The posterior probability and (B) MI of an MPC with hierachically nested spatial periods. Each row of (A) shows posterior distribution for different σ_e's with fixed sensing noise $\sigma^{sen} = 0.1$, with ML (top) and NN (bottom) decoders. NN decoding produces more frequent threshold errors, and a corresponding loss of information, for narrow tuning widths, $\sigma_e \leq 0.01$. These threshold errors of NN results in a sharp decrease in MI as σ_e decreases. This is in contrast to the MI of ML which increases and then saturates as σ_e decreases. Except for the choice of the spatial periods λ's, parameters for numerical simulations are the same as those used in Results. The coding range $[-192\ 192]$ ($R_l = 384$) is quantized into $H = 19200$ bins with equal size $\frac{1}{64}$. For the grid networks, $N = 8, \lambda_n = \{3, 6, 12, 24, 48, 96, 192, 384\}$, $M = 256$, and $r_m \Delta t = 1$. σ^e is varied from 0.0125 to 0.2828 by a factor of $\sqrt{2}$ while $\sigma^{sen} = 0.1$ is fixed.

Appendix B

Supplementary information for Chapter 3

B.1 The product of two Gaussian distributions

The product of two Gaussians $\mathcal{N}(\mu_1, \nu_1)$ and $\mathcal{N}(\mu_2, \nu_2)$ is a scaled Gaussian.

$$\mathcal{N}(\mu_1, \nu_1)\mathcal{N}(\mu_2, \nu_2) = \kappa_{12}\mathcal{N}(\mu_{12}, \nu_{12}) \tag{B.1}$$

where the mean μ_{12}, variance ν_{12}, and scale κ_{12} are given as follows:

$$\mu_{12} = (\frac{\mu_1}{\nu_1} + \frac{\mu_2}{\nu_2})\left(\frac{1}{\nu_1} + \frac{1}{\nu_2}\right)^{-1} \tag{B.2}$$

$$\nu_{12} = \left(\frac{1}{\nu_1} + \frac{1}{\nu_2}\right)^{-1} \tag{B.3}$$

$$\kappa_{12} = \sqrt{\frac{\nu_1\nu_2}{\nu_{12}}}\exp\left(\frac{1}{2}\left\{\frac{\mu_1^2}{\nu_1}\left(\frac{\nu_{12}}{\nu_1} - 1\right) + \frac{\mu_2^2}{\nu_2}\left(\frac{\nu_{12}}{\nu_2} - 1\right) + 2\frac{\nu_{12}}{\nu_1\nu_2}\mu_1\mu_2\right\}\right) \tag{B.4}$$

B.2 A lower bound of the mean square error.

A lower bound on the achievable mean square error (MSE) is derived assuming additional knowledge about the correct set of Gaussians. Under this assumption, the decoder only needs to combine N independent measurements with additive variances $\frac{\sigma^2}{a_i^2}$ $i = 1, 2, \ldots N$. Thus,

$$E\left[(\hat{S} - S)^2\right] \geq \frac{\sigma^2}{\sum_{i=1}^{N} a_n^2}. \tag{B.5}$$

Appendix C

Supplementary information for Chapter 4

C.1 Calculating the probability of threshold error using the union bound

The probability of threshold error is calculated as follows. Suppose that true codeword $\mathbf{X}$ is in the i'th segment and let E_{ij} be the event that this codeword is decodeed to another segment j by noise. Corresponding probability P_{ij} is as follows:

$$P_{ij} = 1 - \Phi\left(\frac{d_{ij}}{2\sigma}\right), \tag{C.1}$$

where d_{ij} is the distance between segments i and j and $\Phi(x)$ is the cumulative distribution of the standard normal distribution defined by $\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-\frac{t^2}{2}}dt$.

Then, the threshold error event is the union of E_{ij} with j being neighbors of i. Thus, the probability of threshold error is

$$P_{th} = P\left(\bigcup_{j\in N(i)} E_{ij}\right), \tag{C.2}$$

where $N(i)$ represents the neighbors of i excluding i. Considering the union bound of (C.2), we have an upper bound on P_{th} with the union replaced by the summation of corresponding probabilities:

$$P_{th} \leq \sum_{j\in N(i)} P_{ij} \leq K\left(1 - \Phi\left(\frac{d_{min}}{2\sigma}\right)\right) \tag{C.3}$$

where K is the number of neighbors and d_{min} is the minimum distance. When the dimension is high, this upper bound is known to be a tight approximation of the threshold error [37].

C.2 The lower bound of distortion of the shift-map codes for a given parameter α.

A lower bound on the distortion of the shift-map code with parameter α is calculated as follows. Since the first encoded variable $X_1 = S$ is most sensitive to the additive noise Z_1, let us consider the case where a large error occurs in the estimate of S because the observation $\mathbf{Y}$ is pushed toward to a different segment of the code segments in the first dimension. To be specific, when $|Z_1| > \frac{1}{2\alpha}$ and $Z_n = 0$ for $n > 1$, the estimation error is $\frac{1}{\alpha}$. In addition, if $|Z_1| > \frac{1}{2\alpha}$ and $Z_n (n > 1)$ have the same sign as Z_1, the estimation error only increases. The set of such $\mathbf{Z}$ is denoted as $\mathcal{T}_1 \subset \mathcal{T}$. The distortion considering only $\mathbf{Z} \in \mathcal{T}_1$ is a lower bound of the distortion considering all $\mathbf{Z} \in \mathcal{T}$. Therefore, the second term in (4.7) is greater than $\text{erfc}\left(\frac{1}{2\sqrt{2}\alpha\sigma}\right)\left(\frac{1}{2}\right)^{(N-2)}\frac{1}{\alpha^2}$, where erfc is the complementary error function. Consequently, the distortion of the shift-map code with α is bounded from below by:

$$D \geq \left(1 - P_{th}^{U}\right)\frac{\sigma^2}{(L^{SM}(\alpha))^2} + \text{erfc}\left(\frac{1}{2\alpha\sigma}\right)\left(\frac{1}{2}\right)^{(N-2)}\frac{1}{\alpha^2}, \tag{C.4}$$

where P_{th}^{U} is the union bound of the probability of threshold error calculated similarly to (C.3), $L^{SM}(\alpha)$ is the stretch factor defined in (4.12).

Bibliography

[1] Grid cell data, http://www.ntnu.edu/kavli/research/grid-cell-data.

[2] L. F. Abbott and Peter Dayan. The effect of correlated variability on the accuracy of a population code. *Neural Computation*, 11(1):91–101, 1999.

[3] Milton Abramowitz and Irene A. Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables,*. U.S. Govt. Print. Off., Washington, 1964.

[4] P. Andersen. *The Hippocampus Book*. Oxford Neuroscience Series. Oxford University Press, USA, 2007.

[5] Erdal Arikan. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *Information Theory, IEEE Transactions on*, 55(7):3051–3073, 2009.

[6] M Baake, U Grimm, and D H Warrington. Some remarks on the visible points of a lattice. *Journal of Physics A: Mathematical and General*, 27(8), 1994.

[7] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, 1996.

[8] F. Barsi and P. Maestrini. Error correcting properties of redundant residue number systems. *Computers, IEEE Transactions on*, C-22(3):307–315, Mar. 1973.

[9] M. Bayati, D. Shah, and M. Sharma. Max-product for maximum weight matching: Convergence, correctness, and LP duality. *Information Theory, IEEE Transactions on*, 54(3):1241–1251, 2008.

[10] Philipp Berens, Alexander S. Ecker, Sebastian Gerwinn, Andreas S. Tolias, and Matthias Bethge. Reassessing optimal neural population codes with neurometric functions. *Proceedings of the National Academy of Sciences*, 2011.

[11] Elwyn R Berlekamp, Robert J McEliece, and Henk CA Van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3):384–386, 1978.

[12] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error-correcting coding and decoding: Turbo-codes. 1. In *Communications, 1993. ICC 93. Geneva. Technical Program, Conference Record, IEEE International Conference on*, volume 2, pages 1064 –1070 vol.2, May 1993.

[13] M. Bethge, D. Rotermund, and K. Pawelzik. Optimal short-term population coding: When fisher information fails. *Neural Computation*, 14(10):2317–2351, 2002.

[14] Raj Chandra Bose and Dwijendra K Ray-Chaudhuri. Further results on error correcting binary group codes. *Information and Control*, 3(3):279–290, 1960.

[15] Raj Chandra Bose and Dwijendra K Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and control*, 3(1):68–79, 1960.

[16] Christian Boucheny, Nicolas Brunel, and Angelo Arleo. A continuous attractor network model without recurrent excitation: Maintenance and integration in the head direction cell system. *Journal of Computational Neuroscience*, 18(2):205–227, 2005.

[17] W. Michael Brown and Alex Bäcker. Optimal neuronal tuning for finite stimulus spaces. *Neural Computation*, 18(7):1511–1526, 2006.

[18] Nicolas Brunel and Jean-Pierre Nadal. Mutual information, fisher information, and population coding. *Neural Computation*, 10(7):1731–1757, 1998.

[19] Yoram Burak and Ila R. Fiete. Accurate path integration in continuous attractor network models of grid cells. *PLoS Comput Biol*, 5(2), Feb 2009.

[20] Yoram Burak and Ila R. Fiete. Fundamental limits on persistent activity in networks of noisy neurons. *Proceedings of the National Academy of Sciences*, 109(43):17645–17650, 2012.

[21] Neil Burgess. Spatial memory: how egocentric and allocentric combine. *Trends in Cognitive Sciences*, 10(12):551 – 557, 2006.

[22] Y Burnod and H Korn. Consequences of stochastic release of neurotransmitters for network computation in the central nervous system. *Proceedings of the National Academy of Sciences*, 86(1):352–356, 1989.

[23] Stephen Cannon, David Robinson, and Shihab Shamma. A proposed neural network for the integrator of the oculomotor system. *Biological Cybernetics*, 49(2):127–136, 1983.

[24] Stephen C. Cannon and David A. Robinson. An improved neural-network model for the neural integrator of the oculomotor system: More realistic neuron behavior. *Biological Cybernetics*, 53(2):93–108, 1985.

[25] B. Chen and G.W. Wornell. Analog error-correcting codes based on chaotic dynamical systems. *Communications, IEEE Transactions on*, 46(7):881–890, July 1998.

[26] S.-Y. Chung. *On the Construction of Some Capacity-Approaching Coding Schemes*. PhD thesis, M.I.T., 2000.

[27] Seong Taek Chung and A. Goldsmith. Degrees of freedom in adaptive modulation: a unified view. In *Vehicular Technology Conference, 2001. VTC 2001 Spring. IEEE VTS 53rd*, volume 2, pages 1267–1271, 2001.

[28] Albert Compte, Nicolas Brunel, Patricia S. Goldman-Rakic, and Xiao-Jing Wang. Synaptic mechanisms and network dynamics underlying

spatial working memory in a cortical network model. *Cerebral Cortex*, 10(9):910–923, 2000.

[29] J. H. Conway, N. J. A. Sloane, and E. Bannai. *Sphere Packings, Lattices, and Groups*. Springer, 3rd edition, 1999.

[30] T. Cover, A.E. Gamal, and M. Salehi. Multiple access channels with arbitrarily correlated sources. *Information Theory, IEEE Transactions on*, 26(6):648–657, Nov. 1980.

[31] Harald Cramér. *Mathematical methods of statistics*. Princeton University Press, Princeton, 1946.

[32] Peter Dayan and L. F. Abbott. *Theoretical neuroscience : computational and mathematical modeling of neural systems*. Massachusetts Institute of Technology Press, Cambridge, Mass., 2001.

[33] A. Destexhe, M. Rudolph, J. M. Fellous, and T. J. Sejnowski. Fluctuating synaptic conductances recreate in vivo-like activity in neocortical neurons. *Neuroscience*, 107(1):13–24, 2001.

[34] Pier Luigi Dragotti and Michael Gastpar. *Distributed Source Coding: Theory, Algorithms and Applications*. Academic Press, 2009.

[35] A. Aldo Faisal, Luc P. J. Selen, and Daniel M. Wolpert. Noise in the nervous system. *Nat Rev Neurosci*, 9(4):292–303, 2008.

[36] Ila R. Fiete, Yoram Burak, and Ted Brookings. What grid cells convey about rat location. *Journal of Neuroscience*, 28(27):6858–6871, 2008.

[37] Jr. Forney, G.D. and G. Ungerboeck. Modulation and coding for linear gaussian channels. *Information Theory, IEEE Transactions on*, 44(6):2384 –2415, 1998.

[38] Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. *Science*, 315:972–976, 2007.

[39] Mark C. Fuhs and David S. Touretzky. A spin glass model of path integration in rat medial entorhinal cortex. *The Journal of Neuroscience*, 26(16):4266–4276, 2006.

[40] Robert Gallager. Low-density parity-check codes. *Information Theory, IRE Transactions on*, 8(1):21–28, 1962.

[41] GallagerR.G. *Low-Density Parity-Check Codes*. PhD thesis, M.I.T, 1963.

[42] M. Gastpar. Uncoded transmission is exactly optimal for a simple gaussian sensor network. *Information Theory, IEEE Transactions on*, 54(11):5247–5251, Nov. 2008.

[43] M. Gastpar, B. Rimoldi, and M. Vetterli. To code, or not to code: lossy source-channel communication revisited. *Information Theory, IEEE Transactions on*, 49(5):1147 – 1158, May 2003.

[44] AP Georgopoulos, AB Schwartz, and RE Kettner. Neuronal population coding of movement direction. *Science*, 233(4771):1416–1419, 1986.

[45] Inmar E. Givoni and Brendan J. Frey. A binary variable model for affinity propagation. *Neural Computation*, 21(6):1589–1600, Nov. 2009.

[46] A. Goldsmith. *Wireless Communications*. Cambridge University Press, 2005.

[47] Ankit Gupta and Sergio Verdú. Nonlinear sparse-graph codes for lossy compression. *IEEE Transactions on Information Theory*, 55(5):1961–1975, 2009.

[48] Torkel Hafting, Marianne Fyhn, Sturla Molden, May-Britt Moser, and Edvard I. Moser. Microstructure of a spatial map in the entorhinal cortex. *Nature*, 436:801–806, 2005.

[49] Kenneth D. Harris, Jozsef Csicsvari, Hajime Hirase, George Dragoi, and Gyorgy Buzsaki. Organization of cell assemblies in the hippocampus. *Nature*, 424(6948):552–556, 2003.

[50] B Hille. *Ion channels of excitable membranes*. Sinauer, 2001.

[51] Alexis Hocquenghem. Codes correcteurs d'erreurs. *Chiffres*, 2(2):147–56, 1959.

[52] G. R. Holt, W. R. Softky, C. Koch, and R. J. Douglas. Comparison of discharge variability in vitro and in vivo in cat visual cortex neurons. *Journal of Neurophysiology*, 75(5):1806–1814, 1996.

[53] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *J Physiol.*, 160(1):106–154, 1962.

[54] H. Krishna, K.-Y. Lin, and J.-D. Sun. A coding theory approach to error control in redundant residue number systems. i. theory and single error correction. *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, 39(1):8–17, Jan. 1992.

[55] Choongkil Lee, William Rohrer, and David Sparks. Population coding of saccadic eye movements by neurons in the superior colliculus. *Nature*, 332(6162):357–360, 1988.

[56] Jill K. Leutgeb, Stefan Leutgeb, May-Britt Moser, and Edvard I. Moser. Pattern separation in the dentate gyrus and ca3 of the hippocampus. *Science*, 315(5814):961–966, 2007.

[57] Stefan Leutgeb, Jill K. Leutgeb, Alessandro Treves, May-Britt Moser, and Edvard I. Moser. Distinct ensemble codes in hippocampal areas ca3 and ca1. *Science*, 305(5688):1295–1298, 2004.

[58] S. Lin and D.J. Costello. *Error Control Coding*. Pearson-Prentice Hall, 2004.

[59] D. Marr. Simple memory: A theory for archicortex. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, 262(841):23–81, 1971.

[60] Alexander Mathis, Andreas V. M. Herz, and Martin Stemmler. Optimal population codes for space: Grid cells outperform place cells. *Neural Computation*, 24(9):2280–2317, 2012.

[61] Alexander Mathis, Andreas V. M. Herz, and Martin B. Stemmler. Resolution of nested neuronal representations can be exponential in the number of neurons. *Phys. Rev. Lett.*, 109:018103, Jul 2012.

[62] MATLAB. *version 7.14.0.739 (R2012a)*. The MathWorks Inc., 2012.

[63] R.J. McEliece, D.J.C. MacKay, and Jung-Fu Cheng. Turbo decoding as an instance of pearl's belief propagation algorithm. *Selected Areas in Communications, IEEE Journal on*, 16(2):140 –152, Feb. 1998.

[64] Bruce L. McNaughton, Francesco P. Battaglia, Ole Jensen, Edvard I Moser, and May-Britt Moser. Path integration and the neural basis of the 'cognitive map'. *Nat Rev Neurosci*, 7(8):663–678, 2006.

[65] Marcelo A. Montemurro and Stefano Panzeri. Optimal tuning widths in population coding of periodic variables. *Neural Computation*, 18(7):1555–1576, 2006.

[66] Edvard I. Moser, Emilio Kropff, and May-Britt Moser. Place cells, grid cells, and the brain's spatial representation system. *Annual Review of Neuroscience*, 31(1):69–89, 2008.

[67] John O'Keefe and Lynn Nadel. The hippocampus as a cognitive map. *Behavioral and Brain Sciences*, 2(4):487–494, 1979.

[68] R. Olfati-Saber, J.A. Fax, and R.M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[69] M. A. Paradiso. A theory for the use of visual orientation information which exploits the columnar structure of striate cortex. *Biological Cybernetics*, 58(1):35–49, 1988.

[70] Anitha Pasupathy and Charles E. Connor. Population coding of shape in area v4. *Nat Neurosci*, 5(12):1332–1338, 2002.

[71] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Representation and Reasoning Series. Morgan Kaufmann, 1988.

[72] Alexandre Pouget, Sophie Deneve, Jean-Christophe Ducom, and Peter E. Latham. Narrow versus wide tuning curves: What's best for a population code? *Neural Computation*, 11(1):85–90, 1999.

[73] R. Quian Quiroga, G. Kreiman, C. Koch, and I. Fried. Sparse but not grandmother-cell coding in the medial temporal lobe. *Trends in cognitive sciences*, 12(3):87–91, 2008.

[74] Irving S Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial & Applied Mathematics*, 8(2):300–304, 1960.

[75] T. Richardson and R.L. Urbanke. *Modern Coding Theory*. Cambridge University Press, 2008.

[76] N. Santhi and A. Vardy. Analog codes on graphs. In *Information Theory, 2003. Proceedings. IEEE International Symposium on*, page 13, 2003.

[77] H S Seung and H Sompolinsky. Simple models for reading neuronal population codes. *Proceedings of the National Academy of Sciences of the United States of America*, 90(22):10749–10753, 1993.

[78] Michael N. Shadlen and William T. Newsome. The variable discharge of cortical neurons: Implications for connectivity, computation, and information coding. *The Journal of Neuroscience*, 18(10):3870–3896, 1998.

[79] Maoz Shamir and Haim Sompolinsky. Nonlinear population codes. *Neural Computation*, 16(6):1105–1136, 2004.

[80] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.

[81] C E Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, Jan. 1949.

[82] Ori Shental, Paul H. Siegel, Jack K. Wolf, Danny Bickson, and Danny Dolev. Gaussian belief propagation solver for systems of linear equations. In *IEEE Int. Symp. on Inform. Theory*, 2008.

[83] D. Slepian and J.K. Wolf. Noiseless coding of correlated information sources. *Information Theory, IEEE Transactions on*, 19(4):471–480, 1973.

[84] M.A. Soderstrand, F.J. Taylor, W.K. Jenkins, and G.A. Jullien. *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*. IEEE PRESS Reprint Series. IEEE, 1986.

[85] William R. Softky and Christof Koch. Cortical cells should fire regularly, but do not. *Neural Computation*, 4(5):643–646, 1992.

[86] WR Softky and C Koch. The highly irregular firing of cortical cells is inconsistent with temporal integration of random epsps. *The Journal of Neuroscience*, 13(1):334–350, 1993.

[87] Haim Sompolinsky, Hyoungsoo Yoon, Kukjin Kang, and Maoz Shamir. Population coding in neuronal systems with correlated noise. *Phys. Rev. E*, 64(5):051904, Oct 2001.

[88] R. Soundararajan and S. Vishwanath. Hybrid coding for gaussian broadcast channels with gaussian sources. In *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*, pages 2790 –2794, Jun. 2009.

[89] Sameet Sreenivasan and Ila Fiete. Grid cells generate an analog error-correcting code for singularly precise neural computation. *Nature Neuroscience*, pages 1330–1337, 2011.

[90] Hanne Stensola, Tor Stensola, Trygve Solstad, Kristian Froland, May-Britt Moser, and Edvard I. Moser. The entorhinal grid map is discretized. *Nature*, 492(7427):72–78, 2012.

[91] Charles F. Stevens. Quantal release of neurotransmitter and long-term potentiation. *Cell*, 72, Supplement(0):55–63, 1993.

[92] M. Taherzadeh and A.K.K. Khandani. Single-sample robust joint source: Channel coding: Achieving asymptotically optimum scaling of SDR versus SNR. *Information Theory, IEEE Transactions on*, 58(3):1565–1577, Mar. 2012.

[93] JS Taube, RU Muller, and JB Ranck. Head-direction cells recorded from the postsubiculum in freely moving rats. i. description and quantitative analysis. *The Journal of Neuroscience*, 10(2):420–435, 1990.

[94] V.A. Vaishampayan and S.I.R. Costa. Curves on a sphere, shift-map dynamics, and error control for continuous alphabet sources. *Information Theory, IEEE Transactions on*, 49(7):1658–1672, July 2003.

[95] R.W. Watson and C.W. Hastings. Self-checked computation using residue arithmetic. *Proceedings of the IEEE*, 54(12):1920–1931, 1966.

[96] Y. Weiss and W.T. Freeman. On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *Information Theory, IEEE Transactions on*, 47(2):736–744, 2001.

[97] Peter E. Welinder, Yoram Burak, and Ila R. Fiete. Grid cells: The position code, neural network models of activity, and the problem of learning. *Hippocampus*, 18(12):1283–1300, 2008.

[98] MA Wilson and BL McNaughton. Dynamics of the hippocampal ensemble code for space. *Science*, 261(5124):1055–1058, 1993.

[99] Kong-Fatt Wong, Alexander C Huk, Michael N Shadlen, and Xiao-Jing Wang. Neural circuit dynamics underlying accumulation of time-varying evidence during perceptual decision making. *Frontiers in Computational Neuroscience*, 1(6), 2007.

[100] A. Wyner. On source coding with side information at the decoder. *Information Theory, IEEE Transactions on*, 21(3):294–300, May 1975.

[101] A. Wyner and J. Ziv. The rate-distortion function for source coding with side information at the decoder. *Information Theory, IEEE Transactions on*, 22(1):1–10, Jan. 1976.

[102] S.S.-S. Yau and Yu-Cheng Liu. Error correction in redundant residue number systems. *Computers, IEEE Transactions on*, 22(1):5–11, 1973.

[103] Y. Yoo, O. O. Koyluoglu, S. Vishwanath, and I. Fiete. Dynamic shift-map coding with side information at the decoder. In *IEEE 50th Allerton Conference on Communication, Control, and Computing*, Monticello, Illinois, USA, Oct. 2012.

[104] MP Young and S Yamane. Sparse population coding of faces in the inferotemporal cortex. *Science*, 256(5061):1327–1331, 1992.

[105] Kechen Zhang and Terrence J. Sejnowski. Neuronal tuning: To sharpen or broaden? *Neural Computation*, 11(1):75–84, 1999.

Vita

Yong Seok Yoo is a doctoral candidate in the department of Electrical and Computer Engineering (ECE) at the University of Texas (UT) at Austin. He received M.S. (Feb. 2005) and B.S. (Aug. 2003) degrees in Electrical Engineering from Seoul National University, with scholarship from Moojin Science and Technology Foundation. He studied for a Ph.D. as a Fulbright scholar, funded by both the U.S. and Korea. He worked as a Graduate Research Assistant (Fall 2010 - Spring 2013, Spring 2014) for Fiete lab with fundings from NSF (IIS-1148973) and DoD-Navy (N00014-13-1-0529). He worked as a Teaching Assistant for the ECE department (Fall 2013). He also served as a Graduate Mentor through Graduates Linked with Undergraduates in Engineering (Spring 2012) and Intellectual Entrepreneurship (Fall 2012). Prior to the doctoral program, he have worked for Samsung Advanced Institute of Technology (2005 - 2008).

During his doctoral study, he received the following awards and honors: Excellence Award from Hyundai Motor Company (2012), Poster Award from Korean-American Scientists and Engineers Association (2012), AKN Research Award from Association of Korean Neuroscientists (2010), Gatsby Cosyne Fellowship from Gatsby Computational Neuroscience Unit, University College London (2010), Academic Competitive Note from the UT at Austin (2009),

and Graduate Dean's Prestigious Fellowship Supplement Fellowship from the UT at Austin (2009).

His research interests include coding theory, graphical models, and network information theory.

Permanent address: 45-103, 313, Apgujeong-ro, Gangnam-gu,
Seoul 135-904, Korea

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.