IMPERIAL

MEng Individual Project

Imperial College London

Department of Computing

THE COMPUTATIONAL ROLE OF COMPLEX REPRESENTATIONS IN RNNS

Author:
Pratyaksh Sharma

Supervisors:
Alexandra Proca
Dr. Pedro Mediano

Second Marker:
Dr. Thomas Heinis

June 13, 2025

Abstract

Our understanding of the internal mechanisms by which neural networks represent and operate on their inputs is very poor, in part due to polysemantic neurons that respond to multiple unrelated features of the input. The *superposition hypothesis* posits that this occurs because models learn to represent more features than they have neurons. Most research on superposition focuses on feed-forward models and the implications of this hypothesis have not been extended to more complex architectures, such as RNNs.

We investigate the internal representations of RNNs from the perspective of superposition. Beginning with an exact analytical decomposition of the expected loss on a recall task, we develop a theoretical framework that predicts and explains the geometric arrangement of features in the hidden states of both linear and non-linear RNNs. We introduce a key distinction between projection interference and composition interference and use this to explain why, under high sparsity, the optimal strategy for non-linear models is to pack most of the task-relevant features into just half of the activation space.

Acknowledgements

I would like to thank my supervisors, Alexandra Proca and Dr. Pedro Mediano, for their insightful guidance and unwavering support. The research process is far from straightforward, full of twists and turns and more dead ends than are typically admitted; the expertise, patience and understanding that Alex and Pedro brought to this project were instrumental to its success.

To my friends: thank you for being the highlight of my time at university. Thank you for laughing with me, supporting me and, at times, putting up with me.

I will always be grateful to my parents and family for striving to give me opportunities beyond my background. To even be here is remarkable and for that I will always be indebted to them.

Contents

List of Figures

Chapter 1

Introduction

Artificial neural networks have achieved widespread success on complex tasks, and yet the internal mechanisms learnt by these models remain very poorly understood. The ability to interpret, at the level of individual neurons or groups of neurons, *how* neural networks represent and operate on their input data would be profound: it would enable us to explain *why* a model made a particular prediction, potentially unlocking a better understanding of the model architecture, the task and of learning itself.

The development of such a fine-grained, mechanistic understanding has been significantly hampered by the fact that information and computation in neural networks often appear to be distributed across many neurons. For example, by analysing the activations of neurons in a vision model, [1] found a neuron that clearly detects the presence of a car in the input image. One might expect that subsequent layers of the model contained a small "circuit" of dedicated neurons for further processing of the car feature but, in actuality, the neuron's activation was spread across many neurons that each appeared to be primarily responsible for something unrelated, such as detecting the presence of a dog. These *polysemantic* neurons that respond to mixtures of unrelated features seem ubiquitous in neural networks of all kinds: [2] found a neuron in a language model that activates in response to academic citations, Korean text and HTTP requests. Although neurons are the fundamental unit of model architectures, they do not appear to be the natural unit for interpretability.

The *superposition hypothesis* [3] suggests that this polysemanticity is not the result of an accidental misalignment between features and neurons, but an emergent compression strategy. In tasks that require a general-purpose model of the world, such as object detection or language modelling, features of the input are inherently very *sparse*: most things rarely occur and unrelated things almost never co-occur. For example, most tokens in general text data are not part of an academic citation or an HTTP request and it is extremely unlikely for a token to be part of both. The hypothesis posits that models learn to take advantage of this property by ignoring the extremely rare cases of co-occurrence and using the space freed up by doing so to represent more, unrelated features [4]. This naturally leads to a situation in which the model represents more features than it has neurons, so each neuron represents more than one feature and individual features are often partially represented by multiple neurons, thus explaining the polysemanticity observed in practice.

Some initial work has been done to study superposition in feed-forward models using simple, small "toy" models and tasks that provide a proof-of-concept setting for interpretability [3, 5, 6]. This line of research has not yet been thoroughly extended to more complex model architectures.

In this work, we study superposition and related concepts in recurrent neural networks (RNNs) using toy models and a simple sequential recall task. This is a more challenging setting than feed-forward models as it introduces a new, fundamentally different axis to the data: time. Moreover, unlike other architectures, an RNN is a dynamical system, meaning that its future behaviour depends on current state; this substantially complicates the analysis.

Our approach is to isolate the temporal axis and focus primarily on superposition between separate activations of a single feature through time. This has the benefit of removing all interactions between different *spatial* features, meaning that our findings can be attributed to the superposition of *temporal* features alone. We study the k-delay task, in which the model must reproduce the input sequence delayed by k time steps. Since the model is not required to transform the input sequence other than shifting it through time, this task is a pure test of memory – a fundamental aspect of real-world sequential processing tasks.

Contributions

The main contribution of this work is a self-consistent and predictive theoretical framework for understanding temporal superposition. In particular, **for linear models**:

- We refine the concept of *interference* introduced in [3] into two distinct phenomena: projection interference and composition interference (section 3.1.3). We argue that these are fundamentally separate concepts and show analytically and empirically that they contribute very differently to the loss and have different geometric implications (section 3.2.4).
- We provide an exact decomposition of the expected loss of an arbitrary linear model on the k-delay task, splitting it into the sum of four geometrically interpretable terms that corresponding to task error, projection interference, composition interference and a mean-correction term (section 3.2.3).
- We prove that linear models that maintain finite mean-squared error as the input sequence length goes to infinity must satisfy the echo state property [7], and find that in two dimensions, the optimal geometric arrangement of features is a *spiral sink* (section 3.3.2).
- We provide detailed analyses of the mean-correction term and composition interference in linear models, explaining why they arise in cases of input data with non-zero mean and become negligible as sparsity is increased (section 3.3.3, section 3.3.4).

We then extend our approach to **non-linear models**:

- We prove an approximation to the k-delay loss for models with linear recurrence and non-linear (ReLU) read-out and verify that this holds in practice.
- We argue that the ReLU read-out creates a half-space in activation space that becomes completely free of all interference in the extremely sparse regime. We hypothesise that, for sufficiently sparse inputs, models will learn to pack as many of their large activations as possible into this half-space.
- We validate our hypothesis empirically and discover a phase transition in the optimal geometric arrangement of features between the dense and sparse regime.
- We introduce a non-linear recurrence for which the linear representation hypothesis provably holds in the extremely sparse regime. We hypothesise that this non-linear model should also exploit the interference-free half-space and verify this empirically.
- We show that the non-linear recurrence creates a privileged basis in activation space and provide a geometrical explanation for how the non-linearity contributes to the increased expressivity of the model.

Finally, we present the generalised hypothesis that, based on the findings in this work, there exist reasonable conditions under which **temporal superposition is fundamentally cheaper than spatial superposition**. This provides a promising direction for future research.

Chapter 2

Background

2.1 Theoretical Background

In this section, we first recall the definition of a recurrent neural network and introduce echo state networks. We then explain various ideas from the field of mechanistic interpretability and discuss the findings of early studies of superposition in feedforward neural networks.

2.1.1 Recurrent Neural Networks

Recurrent neural networks (RNNs) are a fundamental class of artificial neural network architectures designed for machine learning tasks that require sequential processing. Their distinguishing feature is a hidden state that evolves through time: at each time step, the hidden state is updated to integrate both new information – the input in the current step – and old information – the previous hidden state; finally, the hidden state is used to produce an output in each step.

We recall the general sequence modelling task and mathematical description of an RNN, as per [8, p.159-160]. For a given sequence of inputs, $\{\boldsymbol{x}_t\}_{t=1\ldots T}$, with all $\boldsymbol{x}_t \in \mathbb{R}^{N_x}$, the model is tasked with predicting the corresponding sequence of target outputs, $\{\boldsymbol{y}_t\}_{t=1\ldots T}$, with all $\boldsymbol{y}_t \in \mathbb{R}^{N_y}$. Within this report, we define an RNN as

$$\begin{aligned} \boldsymbol{h}_t &= \phi_h\left(\boldsymbol{W}_{\mathrm{h}}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{\mathrm{x}}\boldsymbol{x}_t\right), \\ \boldsymbol{y}_t &= \phi_y\left(\boldsymbol{W}_{\mathrm{y}}\boldsymbol{h}_t\right), \end{aligned}$$

where $\boldsymbol{h}_t \in \mathbb{R}^{N_h}$ is the hidden state after time step t; ϕ_h and ϕ_y are non-linear activation functions; and $\boldsymbol{W}_{\mathrm{h}} \in \mathbb{R}^{N_h \times N_h}$, $\boldsymbol{W}_{\mathrm{x}} \in \mathbb{R}^{N_h \times N_x}$ and $\boldsymbol{W}_{\mathrm{y}} \in \mathbb{R}^{N_y \times N_h}$ are the learnt parameters of the model. Note that the initial hidden state, $\boldsymbol{h}_0$, is typically initialised to $\mathbf{0}$.

The recurrent behaviour of an RNN can be visualised as a "rolled" diagram (Figure figure 2.1). This mechanism allows RNNs to perform computation over sequences, leading to their

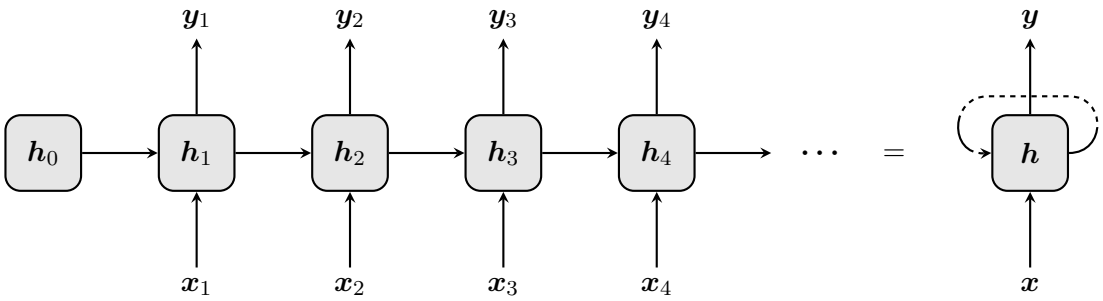


Figure 2.1: "Unrolled" and "rolled" forms of an RNN.

widespread application in natural language processing and time series forecasting [9]. Additionally, RNNs are frequently used in the neuroscience literature to model biological brains, as almost all cortical areas of the brain are recurrently connected to themselves [10].

2.1.2 Echo State Networks, the Echo State Property and Memory Capacity

Echo state networks (ESNs) are a class of RNNs introduced by [7]. Their defining feature is a large, sparsely connected hidden layer with fixed, randomly initialised recurrent weights. The recurrent dynamics are usually non-linear, so even though the recurrent weights are frozen, the output weights alone can be optimised to produce a wide variety of dynamics in the output by reading out from the hidden state in specific directions. This makes the training of ESNs far more efficient than that of RNNs.

Echo state networks are typically initialised to satisfy the *echo state property*, which states that the hidden state $\boldsymbol{h}_t$ should over time converge to a representation of only the input sequence, with the contribution of the initial state approaching zero. Intuitively, this means the initial condition is forgotten over time. For linear systems, such as an RNN with linear recurrence, the echo state property is satisfied if the spectral radius of the recurrent matrix is $\rho(\boldsymbol{W}_\mathrm{h}) < 1$.

Memory capacity is defined by [11] as

$$\mathrm{MC} = \sum_{k=1}^{\infty} \mathrm{MC}_k, \qquad \mathrm{MC}_k = \max_{\boldsymbol{W}_\mathrm{y}} \left[\frac{\mathrm{Cov}^2(\boldsymbol{x}_{t-k}, \boldsymbol{y}_t)}{\mathrm{Var}(\boldsymbol{x}_{t-k})\,\mathrm{Var}(\boldsymbol{y}_t)} \right].$$

Intuitively, the short-term memory capacity MC measures the sum of variance in the input signal that can be recovered in the model output assuming an optimally trained ESN. This corresponds to models being trained on the k-delay task, in which the model must reproduce the input sequence with a delay of k time steps, so $\boldsymbol{y}_t = \boldsymbol{x}_{t-k}$. The authors show analytically that the memory capacity of a linear ESN with an N_h-dimensional hidden state is at most N_h.

2.1.3 Features, Monosemanticity and Polysemanticity

Features

As discussed in [3], the concept of a "feature" is loosely defined in mechanistic interpretability, but its use is motivated by the need to describe the interpretable properties of inputs that neurons respond to. The authors offer three potential definitions:

1. *A feature is any arbitrary function of the input.* This definition is considered mostly unhelpful by [3], as some features appear to form pervasively across models, suggesting they are more fundamental than arbitrary functions.
2. *A feature is a human-interpretable property of the input.* The authors also describe weaknesses in this definition, such as the fact that there may exist genuine, useful features that humans cannot (immediately) interpret. We remark that an example of such a feature may already have been found: the game-winning "move 37" made by AlphaGo [12] was universally unexpected and was initially assumed by experts to be a mistake [13].
3. *A feature is any property of the input that, in a large enough neural network, would reliably be represented by a dedicated neuron.* This is the working definition used by [3], although it is somewhat circular as it relies upon the authors' predictions for representations in a hypothetical "sufficiently large" neural network.

Monosemanticity and Polysemanticity

Monosemanticity is the property of a neuron to respond to only one feature. This makes the behaviour of the neuron easily interpretable [3].

In contrast, polysemanticity is the property of a neuron to respond to multiple unrelated features, such as cat faces and the fronts of cars [14]. Polysemantic neurons are much harder to interpret than monosemantic neurons, as the activation of a polysemantic neuron could be due to any one (or more) of a set of unrelated features [15].

We note that monosemanticity and polysemanticity can be thought of as roughly analogous to the concepts of pure and mixed selectivity in neuroscience, although these concepts were developed independently of each other and in different contexts.

Linear Representation Hypothesis

Motivated by empirical observations and theoretical arguments, the authors of [3] propose the *linear representation hypothesis*, which describes the representational geometry of features by two properties. Firstly, representations are *decomposable*, meaning that they can be described as combinations of independently understandable features of the input; secondly, representations are *linear*, meaning that each feature is represented by a direction in activation space.

The linear representation hypothesis is supported by some findings and contradicted by others. The authors of [3] use word embeddings and the existence of latent spaces in generative models as supportive evidence for linear representation. On the other hand, [16] presents a counterexample to linearity in which features are represented by magnitude rather than direction.

Privileged and Non-Privileged Bases

As described in [3], a privileged basis is one in which the basis directions in activation space are special in some way, due to the network architecture. This most commonly arises when neurons activate according to a non-linear activation function: since the non-linearity is applied to each neuron independently, the action of the non-linearity can only be understood in terms of the standard basis. Consequentially, a privileged basis is one in which the basis directions are encouraged to be interpretable and, conversely, linear representations of features are encouraged to be *basis-aligned*. The authors note that the existence of a privileged basis does not guarantee basis-aligned linear representations of features – it just means there is a reason to pay special attention to the basis directions.

Correspondingly, a non-privileged basis is one in which the basis directions are no more special than any other set of directions. Theoretically, this situation arises in linear network layers, such as word embeddings and the transformer residual stream [3]. In practice, basis-alignment has been observed even in these cases, most likely due to per-dimension normalisation in optimisers such as Adam [17].

2.1.4 Superposition

As per [3], superposition is a strategy for representing n features in fewer than n dimensions; this occurs when the number of features exceeds the number of neurons available to represent them. In such cases, by the pigeonhole principle, at least one neuron must represent more than one feature, so at least one neuron is polysemantic. Superposition can, therefore, be considered a special case of polysemanticity. Note that polysemanticity does not necessitate superposition, as a polysemantic representation of n or fewer features in n neurons can be made monosemantic by a simple change of basis. In superposition, features share dimensions, whether those dimensions correspond to neurons or not.

The superposition hypothesis says that neural networks represent features linearly in almost-orthogonal directions, exploiting two properties of high-dimensional space [3]. Firstly:

Lemma 2.1.1 (Johnson-Lindenstrauss lemma). *Given $0 < \varepsilon < 1$, any n-element subset of a metric space can be embedded in $\mathcal{O}\left(\frac{\log n}{\varepsilon^2}\right)$ dimensions without distorting the pairwise distances between points by more than a factor of $1 \pm \varepsilon$ [18].*

For our purposes, if we choose angular distance as the metric, lemma 2.1.1 implies that $\mathcal{O}\left(e^k\right)$ orthogonal vectors can be embedded *almost-orthogonally* into k-dimensional space. Thus, if the model can tolerate some interference between the representations of features (also referred to as *noise*), it can represent exponentially more features than it could without superposition.

The second property of high-dimensional space exploited by superposition is the ability to perform *compressed sensing*, a technique for recovering a vector from its low-dimensional projection by exploiting the sparsity of the original vector. The hypothesis is that, through these two properties, superposition allows networks to conserve neurons, thus effectively simulating a much larger, sparse model [3, 1].

2.2 Related Work

2.2.1 Investigations of Superposition in Toy Models

The original toy models paper [3] consists of a series of experiments that demonstrate and investigate superposition by training small neural networks on synthetic datasets. The hope is that the phenomena observed in these toy models will lead to intuitions that generalise to much larger, real models. They induce superposition, analyse the regimes in which superposition occurs, investigate the representational geometry and learning dynamics of superposition and link superposition to neural computations. In this section, we recall the methodology and main results of their work in detail, as their approach is highly relevant to our work.

Demonstrating Superposition

Throughout the paper, the authors train a feedforward neural network with a single hidden layer. They train this model as an *autoencoder*, where the model's task is to embed the high-dimensional input vector, $x \in \mathbb{R}^{N_x}$, into a lower-dimensional internal representation, $h \in \mathbb{R}^{N_h}$, and then recover the original vector, x. Their intuition for this approach is based on the superposition hypothesis: if the toy model is simulating a much larger, sparse model, then the the sparse internal representation of the simulated model must be compressed into a much lower-dimensional, superposed internal reprsentation in the toy model, with minimal information loss.

Hence, each element of the input to the toy model is thought of as a hypothetical feature of the simulated model. Since such a simulated model would have sparse internal representations, the authors construct inputs to the toy model of varying sparsity: each element, x_i, of the input is set to 0 with some probability S_i and is otherwise sampled uniformly from $[0, 1]$. In most experiments, the sparsity of all features is set to the same value, S. Additionally, to mimic real data, they assign an importance, I_i, to each feature. In most experiments, $I_i = k^i$ is used, where $0 < k < 1$; this models an exponential *feature importance curve* in which there exist a few key features that are very important for the task, as well as many niche features that are only slightly important. The authors model this by using a feature's importance to determine its contribution to the loss function, defined as

$$L = \sum_x \sum_i I_i \left(x_i - x_i'\right)^2.$$

The authors initially investigate superposition in non-privileged bases, so the hidden layer is linear. The output layer has a bias and uses the ReLU activation function; these are explained to be important for superposition as they enable the model to filter out the noise arising from interference. In particular, the authors note that the inclusion of an activation function is critical to superposition, as a fully linear model would effectively just perform a principal component analysis (PCA). Finally, the output weights are set to the tranpose of the input weights, W; this eliminates any discrepancy between the directions that features are "written" to versus the

directions they are "read" from. This leads to the first toy model:

$$x' = \text{ReLU}\left(W^\top W x + b\right).$$

The authors of [3] interpret each column, W_i, of the input weight matrix as the direction in activity space that represents feature i. The norm, $\|W_i\|$, determines the degree to which feature i is represented by the model. Additionally, the authors define the quantity $\sum_{j\neq i}(\hat{W}_i \cdot W_j)^2$, where $\hat{W}_i$ is the unit W_i vector, as a measure of the extent to which feature i shares its dimension with other features, thus quantifying superposition.

They observe that, in the dense regime, the model represents the N_h most important features orthogonally and ignores all other features. However, as the sparsity of inputs is increased, the model begins to include lower-importance features in its internal representation via superposition; in the high-sparsity regime, all features are represented in superposition.

Superposition as a Phase Change

The authors then investigate the conditions in which a feature is represented in a dedicated dimension, in superposition, or not at all. They train a toy model with two inputs and just one neuron in the hidden layer. The input is varied on two axes: sparsity, as before, and also the relative importance of the second feature with respect to the first. In each setting, they measure the norm and superposition of the second feature.

In the dense regime, the model allocates a dedicated dimension to whichever feature is more important, ignoring the other one. As mentioned above, this is a PCA-like solution. As sparsity is increased, the authors observe a point beyond which the model switches to representing both features in superposition. Due to the discrete nature of this switch from one state to another, the authors call this a *phase change*. Notably, when there is a larger difference in importance between the two features, it takes sparser input to reach this point of phase change.

The authors study the boundaries of phase change by plotting a phase diagram that shows the regions in which each type of solution occurs. They also extend the approach to study a network with three inputs and two hidden layer neurons. Finally, they verify their empirical results with a theoretical prediction of the phase diagram based on analytical calculations of the loss for each kind of representation.

The Geometry of Superposition

In the previous sections, the authors quantified superposition by measuring the extent to which a feature "shared" its dimension with other features. Here, they develop a quantity that measures the "fraction of a dimension" that a feature gets. The *dimensionality* of feature i is given by

$$D_i = \frac{\|W_i\|^2}{\sum_j (\hat{W}_i \cdot W_j)^2},$$

where the numerator represents the extent to which feature i is represented and the denominator measures the number of features that share feature i's embedding dimension. An orthogonally represnted feature has a dimensionality of 1, while a feature in an *antipodal pair* (a pair of features represented by opposite directions) has a dimensionality of $\frac{1}{2}$.

The authors produce a plot that visualises how the distribution of feature dimensionality changes with increased sparsity. Across different sparsity levels, they observe persistent lines of clusters in these distributions at dimensionalities of $\frac{1}{2}$, $\frac{3}{4}$, $\frac{2}{3}$, $\frac{2}{5}$ and $\frac{3}{8}$. They link these to geometry by interpreting these fractional dimensionalities as polytopes: digons (2 features in 1 dimension), tetrahedrons (4 features in 3 dimensions), triangles (3 features in 2 dimensions), pentagons (5 features in 2 dimensions) and square antiprisms (8 features in 3 dimensions). The

authors note that this geometric interpretation may be a quirk of the toy model, rather than a pattern that can be generalised to real models.

The paper also explores the effects of non-uniform superposition on the geometry. In particular, they investigate the effects of correlation and anticorrelation betwen features. To generate correlated synthetic data, they partition the input features into *correlated feature sets*; for each feature set, they use a single binary random variable to determine whether or not its features are all set to 0; if they are not set to 0, they are independently sampled from a uniform distribution over $[0, 1]$ as before. To generate *anticorrelated feature sets*, they use a similar process, but when the features in the set are not all set to 0, only one of them is randomly selected and set to a value uniformly sampled from $[0, 1]$; the other features in the set are kept at 0.

The authors observe fairly intuitive results: correlated features prefer to be orthogonally represented, while anticorrelated features prefer to be represented in opposite directions. When superposition is necessary, correlated features prefer to be represented side-by-side, preferring positive interference. In some cases, where features are denser or more correlated, multiple correlated features are seen to "collapse" into a single direction given by the first principal component of those features; the authors propose a trade-off between PCA and superposition.

More interestingly, the paper discusses the tendency of correlated features in larger models to generate a "local almost-orthogonal basis", where the overall representation is in superposition but there exist sets of features that are almost orthogonal and exhibit very little superposition. They remark that, if this finding is applicable to real networks, we could make the assumption of local non-superposition for certain subdistributions in activation space, which would allow us to apply PCA and potentially identify the underlying features.

Neural Computations in Superposition

The authors suggest that superposition is more than just a compression mechanism and explore how neural networks can perform computation in superposition. They use a modified experimental setup, in which there is a privileged basis and the input weights are separate from the output weights:

$$h = \mathrm{ReLU}\left(W_1 x\right),$$
$$y' = \mathrm{ReLU}\left(W_2 h + b\right).$$

The model is trained to predict $y = \mathrm{abs}(x)$. Accordingly, the synthetic feature input is modified to be sampled uniformly from $[-1, 1]$ when it is non-zero. The authors describe how such a model could compute, without superposition, the absolute value of N_x inputs using $2N_x$ hidden layer neurons: for each input, x_i, one hidden layer neuron calculates $\mathrm{ReLU}(x_i)$, while another calculates $\mathrm{ReLU}(-x_i)$; these values are then added together by an output layer neuron.

The authors demonstrate that neural networks can compute the absolute value of N_x features with fewer than $2N_x$ hidden layer neurons, using superposition. They train a model with 100 features (with an exponential importance curve) and 40 hidden layer neurons. As the representations are in a privileged basis, the authors are able to analyse what each neuron represents.

The results show that in the dense regime, each neuron represents one feature, so all neurons are monosemantic. As sparsity is introduced, some neurons begin to represent a few features of lesser importance, exhibiting polysemanticity, while other neurons continue to represent the most important features monosemantically. Increasing sparsity leads to a mix of neurons: some highly polysemantic neurons that represent many low-importance features, some slightly polysemantic neurons that represent a few mid-importance features and some monosemantic neurons that represent the most important features. We remark that this is effectively a reflection of the feature importance curve, where the model has learnt that representing many low-importance features can be as valuable as representing one or a few higher-importance features. Moreover, this experiment demonstrates that gradient descent is able to find a way to use the ReLU

activation function to compute the absolute value, even when the features exist in superposition across multiple neurons and each neuron represents a mix of features.

Chapter 3

Temporal Superposition in Linear RNNs

RNNs introduce a new dimension to our study of internal representations: time. In this chapter, we study this temporal axis in isolation by restricting our attention to RNNs that process sequences of scalar inputs using linear recurrence. The linear setup allows for an analytical approach where we can write down exact expressions for the loss in terms of feature representations through time; using scalar inputs eliminates the spatial dimensions, allowing us to focus solely on temporal effects.

We begin by describing how the linear representation hypothesis manifests in linear RNNs and introducing the concept of temporal features. We remark that the number of temporal features grows with time, making forgetting or superposition essentially inevitable.

We then introduce a toy task that requires fixed-delay recall and show that the squared-error loss decomposes exactly into the sum of four interpretable incentives. This decomposition makes it clear which features the model benefits from representing.

We track the values of these four terms over a training run; this allows us to interpret the learning dynamics of the model from a geometric perspective. We observe that the loss decreases in stages that correspond to distinct geometric strategies, eventually settling on an arrangement in which features spiral into the origin as they become older.

Building on this result, we study the computational role of the recurrent weight matrix by focusing on special cases in 2 dimensions. We find that optimal fixed-delay recall is achieved by matrices that satisfy the echo state property and, in particular, implement spiral sinks.

Next, we consider the effect of introducing an explicit output bias and show that this recovers a simplified, more easily interpretable expression for the loss in the limit of infinitely long sequences. We show that this is expression is a good approximation of empirically observed loss.

Finally, we provide a geometric explanation of composition interference and discuss the three cases in which it becomes negligible. We explain quirks of this term in the linear setting, such as a preference for tegum products of antipodal pairs.

3.1 Linear Representation of Temporal Features

3.1.1 Unrolling the Hidden State of a Linear RNN

Throughout this chapter, we work with linear RNNs that take a scalar input and produce a scalar output at each time step. In particular, for $t \geq 1$, we define

$$\boldsymbol{h}_0 = \boldsymbol{0}, \qquad \boldsymbol{h}_t = \boldsymbol{W}_{\mathrm{x}} x_t + \boldsymbol{W}_{\mathrm{h}} \boldsymbol{h}_{t-1}, \qquad y_t = \boldsymbol{W}_{\mathrm{y}}^{\top} \boldsymbol{h}_t,$$

where the scalar input and output are $x_t, y_t \in \mathbb{R}$ and hidden state is $\boldsymbol{h}_t \in \mathbb{R}^{N_h}$. The input vector and read-out vector are $\boldsymbol{W}_{\mathrm{x}}, \boldsymbol{W}_{\mathrm{y}} \in \mathbb{R}^{N_h}$ and the recurrent weight matrix is $\boldsymbol{W}_{\mathrm{h}} \in \mathbb{R}^{N_h \times N_h}$. Expanding the recurrence in $\boldsymbol{h}_t$, we obtain

$$\begin{aligned}
\boldsymbol{h}_t &= \boldsymbol{W}_{\mathrm{x}} x_t + \boldsymbol{W}_{\mathrm{h}} \boldsymbol{W}_{\mathrm{x}} x_{t-1} + \boldsymbol{W}_{\mathrm{h}}^2 \boldsymbol{W}_{\mathrm{x}} x_{t-2} + \cdots + \boldsymbol{W}_{\mathrm{h}}^{t-1} \boldsymbol{W}_{\mathrm{x}} x_1 + \boldsymbol{W}_{\mathrm{h}}^t \boldsymbol{h}_0 \\
&= \boldsymbol{W}_{\mathrm{x}} x_t + \boldsymbol{W}_{\mathrm{h}} \boldsymbol{W}_{\mathrm{x}} x_{t-1} + \boldsymbol{W}_{\mathrm{h}}^2 \boldsymbol{W}_{\mathrm{x}} x_{t-2} + \cdots + \boldsymbol{W}_{\mathrm{h}}^{t-1} \boldsymbol{W}_{\mathrm{x}} x_1 \\
&= \sum_{s=0}^{t-1} \boldsymbol{W}_{\mathrm{h}}^s \boldsymbol{W}_{\mathrm{x}} x_{t-s}.
\end{aligned}$$

3.1.2 Temporal Features and Memory Capacity

Defining $\boldsymbol{w}_s = \boldsymbol{W}_{\mathrm{h}}^s \boldsymbol{W}_{\mathrm{x}} \in \mathbb{R}^{N_h}$, we can express the hidden state at time t as

$$\boldsymbol{h}_t = \sum_{s=0}^{t-1} \boldsymbol{w}_s x_{t-s}.$$

Each of the inputs x_{t-s} is independently and linearly represented in the hidden state in the direction given by $\boldsymbol{w}_s$; this motivates our view of the inputs x_{t-s} as *temporal features*.

This perspective makes it clear that, at time t, the model technically has access to the entire history of t inputs as features. Inevitably, for $t > N_h$, the hidden state becomes a bottleneck: it is forced to either drop some features (essentially forgetting them) or represent more features than it has dimensions (in superposition). This is notably different from the feed-forward case: theoretically, for any task, there exists a disentangled feed-forward model large enough to represent every feature, because the number of features is constant; for RNNs, the number of features increments with each time step, so every RNN is eventually forced to choose between forgetting and superposition.

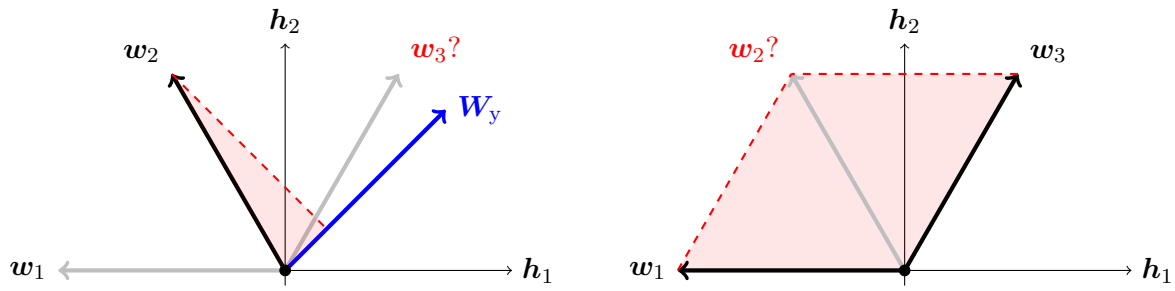
3.1.3 Projection and Composition Interference

Features represented in superposition will interfere with each other. We distinguish between two fundamentally different types of interference that will be crucial in our understanding of superposition in RNNs:

1. **Projection interference:** where the activation of a single feature is mistakenly read out as a slight activation of another. This occurs when a feature is represented non-orthogonally to a read-out vector, causing an unintended non-zero projection onto that read-out.

2. **Composition interference:** where the simultaneous activation of multiple features is linearly combined into an activation that imitates another feature. This occurs due to the trade-off between composition and superposition discussed in [4].

These are illustrated in figure 3.1 on the following page. Note that these two forms of interference differ significantly in terms of when they can arise: projection interference requires the presence



(a) Projection interference

(b) Composition interference

Figure 3.1: The difference between projection and composition interference. In both figures, three features are represented linearly in a 2D activation space by vectors $\boldsymbol{w}_1, \boldsymbol{w}_2$ and $\boldsymbol{w}_3$. In (a), only the vector $\boldsymbol{w}_2$ is active, but its projection onto $\boldsymbol{W}_y$ creates ambiguity: it could equally be explained by a slight activation of $\boldsymbol{w}_3$. For a downstream consumer that only sees the read-out from $\boldsymbol{W}_y$, there is no way to disambiguate these two cases. In (b), both $\boldsymbol{w}_1$ and $\boldsymbol{w}_3$ are active, while $\boldsymbol{w}_2$ is not; nevertheless, the two active vectors combine linearly to form an activation identical to that of $\boldsymbol{w}_2$. The activation of two features have thus inextricably "fused" into an impersonation of an inactive, unrelated feature.

of a read-out vector, while composition interference requires two or more features to be active simultaneously.

We remark that an interesting consequence of composition interference is that it can manifest in RNNs as time-shifting of a single feature. For instance, suppose that the vector $\boldsymbol{w}_1, \boldsymbol{w}_2$ and $\boldsymbol{w}_3$ in figure 3.1(b) are temporal features corresponding to the presence of a single spatial feature over a sequence of three inputs. Then, as shown in the figure, a sequence that activates the spatial feature in the first and third time step could actually be treated in the model as a sequence that activates it only in the *second* time step. Naturally, this effect would then propagate through the model and show up in the output.

3.2 Interpreting the k-Delay Loss

3.2.1 The k-Delay Task

We study the k-delay task, in which the model is trained to reproduce the input sequence after a fixed delay of k time steps. This classic task has been used as a benchmark for short term memory capacity in echo state networks [11]; we use it here because it gives us direct and interpretable control over the memory requirement that an RNN is trained to satisfy.

Concretely, the RNN is trained to produce at each time step $y_t = x_{t-k}$. This can be viewed as a regression task, so we use a squared-error loss to quantify the error at each time step:

$$L^{(t)} = (x_{t-k} - y_t)^2,$$

where, for convenience, we let $x_t = 0$ for $t \leq 0$. Intuitively, the model is trained to output 0 for the first k time steps and then begin producing $x_1, x_2, \ldots$ from time $t = k + 1$ onwards.

3.2.2 Modelling Assumptions

Temporal Independence We want to find the expected value of the squared-error loss defined above. We model the input as an i.i.d. stochastic process of scalar random variables $\{X_t\}_{t\geq 1}$. Of course, in practice, sequential data is rarely independent through time; the value of X_t usually conveys information about the values of X_{t+1}, X_{t+2} and so on. This is, therefore, a strong assumption that we do not expect to hold in practical settings. Nevertheless, we proceed with the i.i.d. input sequence as a simplifying assumption because it makes the k-delay task

more clearly interpretable, simplifies our analytical approach and is a standard assumption made in the study of memory capacity [11, 19].

Temporal Sparsity Following [3], we now let $X_t = B_t U_t$, where $B_t \sim \text{Bernoulli}(p)$, U_t are identically distributed according to any distribution and all $\{B_t\}_{t\geq 1} \cup \{U_t\}_{t\geq 1}$ are mutually independent. This allows us to explicitly control *temporal sparsity* by varying p: smaller p corresponds to rarer feature presence, meaning higher sparsity. Note that this in itself is purely a generalisation – setting $p = 1$ recovers an arbitrary i.i.d. stochastic process.

In practice, the assumption of high temporal sparsity (p close to 0) is very reasonable. As argued by [3], in text sequences, any token activates only a tiny proportion of all possible features; conversely, any feature, such as the mention of a specific city, is very rarely activated. Even biological neurons have been observed to activate in sparse bursts at precise times when producing sequences of birdsong [20].

3.2.3 Decomposing the k-Delay Loss

Define $a_s = \boldsymbol{W}_{\text{y}}^{\top} \boldsymbol{w}_s \in \mathbb{R}$ as the projection of each $\boldsymbol{w}_s$ onto the read-out vector $\boldsymbol{W}_{\text{y}}$. Then we can express the model's output at time t as

$$y_t = \boldsymbol{W}_{\text{y}}^{\top} \sum_{s=0}^{t-1} \boldsymbol{w}_s x_{t-s} = \sum_{s=0}^{t-1} \boldsymbol{W}_{\text{y}}^{\top} \boldsymbol{w}_s x_{t-s} = \sum_{s=0}^{t-1} a_s x_{t-s}.$$

The expected loss incurred by the model at timestep t is then given by

$$\begin{aligned} L^{(t)} &= \mathbb{E}\left[\left(X_{t-k} - \sum_{s=0}^{t-1} a_s X_{t-s}\right)^2\right] \\ &= \mathbb{E}\left[X_{t-k}^2\right] - 2\,\mathbb{E}\left[X_{t-k} \sum_{s=0}^{t-1} a_s X_{t-s}\right] + \mathbb{E}\left[\left(\sum_{s=0}^{t-1} a_s X_{t-s}\right)^2\right]. \end{aligned}$$

Note that for notational convenience,we are assuming temporarily that $t > k$. For $t \leq k$, $X_{t-k} = 0$ as before, so only the third expectation is non-zero. We will address this once we obtain a final expression for $L^{(t)}$ at the end of this section.

We compute each of these expectations separately. Recall that we have defined $X_t = B_t U_t$. Denoting the first and second moments of U_t by $\mu = \mathbb{E}[U_t]$ and $\nu = \mathbb{E}[U_t^2]$, we begin with

$$\mathbb{E}\left[X_{t-k}^2\right] = \mathbb{E}\left[B_{t-k}^2 U_{t-k}^2\right] = \mathbb{E}\left[B_{t-k}^2\right] \mathbb{E}\left[U_{t-k}^2\right] = \mathbb{E}\left[B_{t-k}\right] \mathbb{E}\left[U_{t-k}^2\right] = p\nu.$$

Next, we evaluate the middle term, being careful to handle the case of $s = k$ separately:

$$\begin{aligned} \mathbb{E}\left[X_{t-k} \sum_{s=0}^{t-1} a_s X_{t-s}\right] &= \mathbb{E}\left[X_{t-k} a_k X_{t-k} + X_{t-k} \sum_{\substack{s=0 \\ s\neq k}}^{t-1} a_s X_{t-s}\right] \\ &= a_k\,\mathbb{E}\left[X_{t-k}^2\right] + \sum_{\substack{s=0 \\ s\neq k}}^{t-1} a_s\,\mathbb{E}\left[X_{t-k} X_{t-s}\right] \\ &= p\nu a_k + \sum_{\substack{s=0 \\ s\neq k}}^{t-1} a_s\,\mathbb{E}\left[B_{t-k}\right] \mathbb{E}\left[B_{t-s}\right] \mathbb{E}\left[U_{t-k}\right] \mathbb{E}\left[U_{t-s}\right] \\ &= p\nu a_k + p^2\mu^2 \sum_{\substack{s=0 \\ s\neq k}}^{t-1} a_s. \end{aligned}$$

Finally, we evaluate the third term, being careful to handle the diagonal terms separately:

$$\begin{aligned}\mathbb{E}\left[\left(\sum_{s=0}^{t-1} a_s X_{t-s}\right)^2\right] &= \mathbb{E}\left[\sum_{s=0}^{t-1} a_s^2 X_{t-s}^2 + \sum_{\substack{s_1,s_2=0\\ s_1\neq s_2}}^{t-1} a_{s_1} a_{s_2} X_{t-s_1} X_{t-s_2}\right] \\ &= \sum_{s=0}^{t-1} a_s^2\, \mathbb{E}\left[X_{t-s}^2\right] + \sum_{\substack{s_1,s_2=0\\ s_1\neq s_2}}^{t-1} a_{s_1} a_{s_2}\, \mathbb{E}\left[X_{t-s_1} X_{t-s_2}\right] \\ &= p\nu \sum_{s=0}^{t-1} a_s^2 + p^2\mu^2 \sum_{\substack{s_1,s_2=0\\ s_1\neq s_2}}^{t-1} a_{s_1} a_{s_2}.\end{aligned}$$

Putting these together, we obtain an exact expression for the expected loss at time t:

$$\begin{aligned}L^{(t)} &= \mathbb{E}\left[X_{t-k}^2\right] - 2\,\mathbb{E}\left[X_{t-k}\sum_{s=0}^{t-1} a_s X_{t-s}\right] + \mathbb{E}\left[\left(\sum_{s=0}^{t-1} a_s X_{t-s}\right)^2\right] \\ &= p\nu - 2\left(p\nu a_k + p^2\mu^2 \sum_{\substack{s=0\\ s\neq k}}^{t-1} a_s\right) + p\nu \sum_{s=0}^{t-1} a_s^2 + p^2\mu^2 \sum_{\substack{s_1,s_2=0\\ s_1\neq s_2}}^{t-1} a_{s_1} a_{s_2} \\ &= p\nu + 2p\nu a_k - 2p^2\mu^2 \sum_{s\neq k}^{t-1} a_s + p\nu \sum_{s=0}^{t-1} a_s^2 + p^2\mu^2 \sum_{s_1\neq s_2}^{t-1} a_{s_1} a_{s_2} \\ &= \underbrace{p\nu\,(a_k-1)^2}_{\text{(i)}} + \underbrace{p\nu \sum_{s\neq k}^{t-1} a_s^2}_{\text{(ii)}} - \underbrace{2p^2\mu^2 \sum_{s\neq k}^{t-1} a_s}_{\text{(iii)}} + \underbrace{p^2\mu^2 \sum_{s_1\neq s_2}^{t-1} a_{s_1} a_{s_2}}_{\text{(iv)}}. \qquad (*)\end{aligned}$$

As alluded to before, it is implicit in this expression that terms (i) and (iii) are only present for $t > k$. For $t \leq k$, only terms (ii) and (iv) apply, which – as we will see in section 3.2.4 – are the terms that correspond to interference. We will mainly be concerned with understanding this expression for large or infinite t, as motivated in section 3.3.2.

Perfect delay-line solution For $N_h \geq k+1$ (or greater), the k-delay task is perfectly solved by using the $(k+1)$-dimensional *lower shift matrix* as $\boldsymbol{W}_\text{h}$ and setting $\boldsymbol{W}_\text{x} = \boldsymbol{e}_1$, $\boldsymbol{W}_\text{y} = \boldsymbol{e}_{k+1}$ [19]. We verify that this solution achieves zero loss at each time step. In this arrangement, we have $\boldsymbol{w}_s = \boldsymbol{e}_{s+1}$ for $s \leq k$ and $\boldsymbol{w}_s = \boldsymbol{0}$ for $s > k$. Since $\boldsymbol{W}_\text{y} = \boldsymbol{e}_{k+1}$, we have $a_k = 1$ and $a_s = 0$ for all $s \neq k$. Thus $(*)$ simplifies to $L^{(t)} = p\nu(1-1)^2 + p\nu(0) - 2p^2\mu^2(0) + p^2\mu^2(0) = 0$.

3.2.4 A Geometric Interpretation of Loss Terms

We initially consider the case of $\mu = 0$, where the input distribution is centered at zero. In this case, the expected loss at time t simplifies to

$$L^{(t)} \propto (a_k - 1)^2 + \sum_{s\neq k}^{t-1} a_s^2. \qquad (**)$$

We will refer to this expression as the simple form of the loss. Its two components are:

(i) Task benefit: an incentive to perform the task by producing x_{t-k} at time t. This encourages the vector $\boldsymbol{w}_k = \boldsymbol{W}_\text{h}^k \boldsymbol{W}_\text{x}$, that represents feature x_{t-k}, to align with the read-out direction $\boldsymbol{W}_\text{y}$ and to adjust its norm to keep the projection onto $\boldsymbol{W}_\text{y}$ close to 1.

(ii) Projection interference cost: a penalty on any $\boldsymbol{w}_s$ vector – other than $\boldsymbol{w}_k$ – that has a non-zero projection onto the read-out direction $\boldsymbol{W}_{\text{y}}$. This term encourages these temporal features to be represented orthogonally to $\boldsymbol{W}_{\text{y}}$ or with a small norm (barely represented).

Let us now consider the general case where the input distribution may have a non-zero mean. In this case, two further terms, (iii) and (iv), arise in $L^{(t)}$. Recalling that the RNN being studied has no bias term, we can interpret these two terms in the per-timestep loss as follows:

(iii) Mean correction: an incentive to offset the non-zero mean by carefully aligning some $\boldsymbol{w}_s$ vectors (other than $\boldsymbol{w}_k$) to have non-zero projections onto $\boldsymbol{W}_{\text{y}}$. The existence of this term implies that some projection interference can work in favour of the model, if carefully calibrated – we will see evidence of this in figure 3.2. We will discuss mean correction and its relationship to bias in section 3.3.3.

(iv) Composition interference: the overall effect on the loss of cases where multiple temporal features simultaneously activate. This is neither a pure cost, nor a pure benefit: it may be positive or negative. Geometrically, this term penalises positive correlation of $\boldsymbol{w}_s$ vectors while rewarding negative correlation, with respect to their projection onto $\boldsymbol{W}_{\text{y}}$ – negative (destructive) interference is preferred over positive (constructive) interference. In the language of [3], this term can be seen as a kind of Thomson problem, encouraging the $\boldsymbol{w}_s$ vectors to spread out in activation space as much as possible, ideally forming antipodal pairs. Notably, temporal features represented in a tegum product of antipodal pairs will be purely rewarded by this term. We will discuss composition interference further in section 3.3.4 explaining why it is fundamentally tied to the non-zero input data mean.

Observe that terms (iii) and (iv) are $\mathcal{O}(p^2)$ in $L^{(t)}$ while the other terms are $\mathcal{O}(p)$. This implies that, even for data with a non-zero mean $\mu \neq 0$, in the extremely sparse regime where we take the limit of $p \to 0$, the impact of these two terms on the loss becomes negligible:

- Term (iii) disappears because, even though the underlying U has a fixed mean $\mathbb{E}[U] = \mu$, the mean of the overall input X_t is proportional to its sparsity: $\mathbb{E}[X] = \mathbb{E}[B]\,\mathbb{E}[U] = p\mu$, so $\mathbb{E}[X] \to 0$ as $p \to 0$; extremely sparse data *does* have an approximately zero mean.
- Term (iv) disappears because the probability of two or more temporal features simultaneously activating is $\mathcal{O}(p^2)$, so composition interference becomes negligible as $p \to 0$.

3.2.5 Identifying Task-Relevant and Irrelevant Features

Given this interpretation of the loss, we introduce some useful terminology to distinguish different kinds of temporal features present in the k-delay task at time t:

- **Output feature:** x_{t-k}, represented by the $\boldsymbol{w}_k$ vector. By definition of the task, this is the temporal feature that the model should aim to read out using $\boldsymbol{W}_{\text{y}}$. In ($*$), it is the only feature that contributes to the task error term.
- **Intermediate features:** x_{t-s} for $0 \leq s < k$, represented by the k corresponding $\boldsymbol{w}_s$ vectors. These are temporal features that the model is holding in memory to be read out in future time steps. They do not contribute to the immediate task performance 'as they should not yet appear in the output, but they do contribute to future task performance. This incentivises the model to represent them as faithfully as possible.
- **Task-relevant features:** the $k+1$ output and intermediate features. These are the features that can contribute to task performance either immediately or in the future, so the model should prioritise their representations.

- **Historical (task-irrelevant) features:** x_{t-s} for $k < s < t$, represented by the $t-k-1$ corresponding $\boldsymbol{w}_s$ vectors. These features cannot contribute to current or future task performance, so the only way they can be useful is via the mean correction benefit (if it is present). The model is therefore incentivised to forget these features or, in the case of non-zero mean and sufficiently low sparsity, to use them for mean correction.

Note that we will often loosely refer to the $\boldsymbol{w}_s$ vectors and the a_s projections as "features" even though, technically, they are the vectors and scalar projections that *represent* the features. We will make this distinction explicit when it is not obvious from the context.

3.3 Understanding and Extending the Linear Architecture

3.3.1 A Geometric Interpretation of Learning Dynamics

We have found an exact expression for the expected loss at each time step and decomposed it into interpretable components. As the linear RNN is a linear time-invariant system, its performance depends on the model parameters only through the a_s terms, which are equivalent to the model's impulse response. The $\boldsymbol{w}_s$ vectors therefore contribute to $L^{(t)}$ solely via their projections onto $\boldsymbol{W}_{\text{y}}$, as visualised for a trained RNN in figure 3.2 on the next page. In the remainder of this subsection, we study this empirical result and make a number of informative observations.

Firstly, it is encouraging that the empirical loss closely follows the expected value, providing evidence for the correctness of our decomposed expression. This means we can safely analyse the shape of the expected loss curve rather than the actual curve, which contains far more noise.

An interesting feature of the expected loss curve is that it decreases in stages rather than continuously, similar to the effect observed by [3]. In particular, we identify (by eye) four distinct regimes, labelled in the figure as R1, R2, R3 and R4. The bottom-left plot of a_s values provides a direct connection between a model's performance and its representational geometry, so we can interpret each regime geometrically.

R1: Immediately, the model tries to decrease the loss by aligning all the $\boldsymbol{w}_s$ vectors to correlate positively with $\boldsymbol{W}_{\text{y}}$. This decreases task error and takes advantage of the mean correction term, providing an initial improvement to the loss. All the vectors become positively correlated with each other (facing similar directions), leading to increased interference.

R2: Eventually, the mean interference term plateaus, suggesting that the model has compensated for the non-zero mean. In this regime, the overall loss is steady but we see some oscillatory behaviour in the learning dynamics and eventually a large change in a_0 (and, correspondingly, $\boldsymbol{w}_0$). This is consistent across training runs. We speculate that $\boldsymbol{w}_0$ is separating from the other $\boldsymbol{w}_s$ vectors and rotating through activation space, eventually kick-starting a similar movement in the other vectors that leads to the next regime.

R3: Next, the model transitions into its final geometric arrangement, spreading the $\boldsymbol{w}_s$ vectors out across the entire plane. As part of this, a_k becomes much larger, causing a significant drop in task error; on the other hand, some of the other a_s values also increase in magnitude, causing projection interference to increase. As some of the a_s values fall below zero, the model is less able to take advantage of mean correction, but on the other hand, the even distribution of vectors causes a drop in composition interference.

R4: In the final regime, the model has settled into the arrangement seen in the figure.

There are a few notable features visible in the final arrangement of temporal features. Firstly, we often observe that pairs of $\boldsymbol{w}_s$ vectors face in exactly opposite directions, forming antipodal pairs. This is a natural consequence of the composition interference term in $(*)$ preferring negatively correlated pairs of vectors.

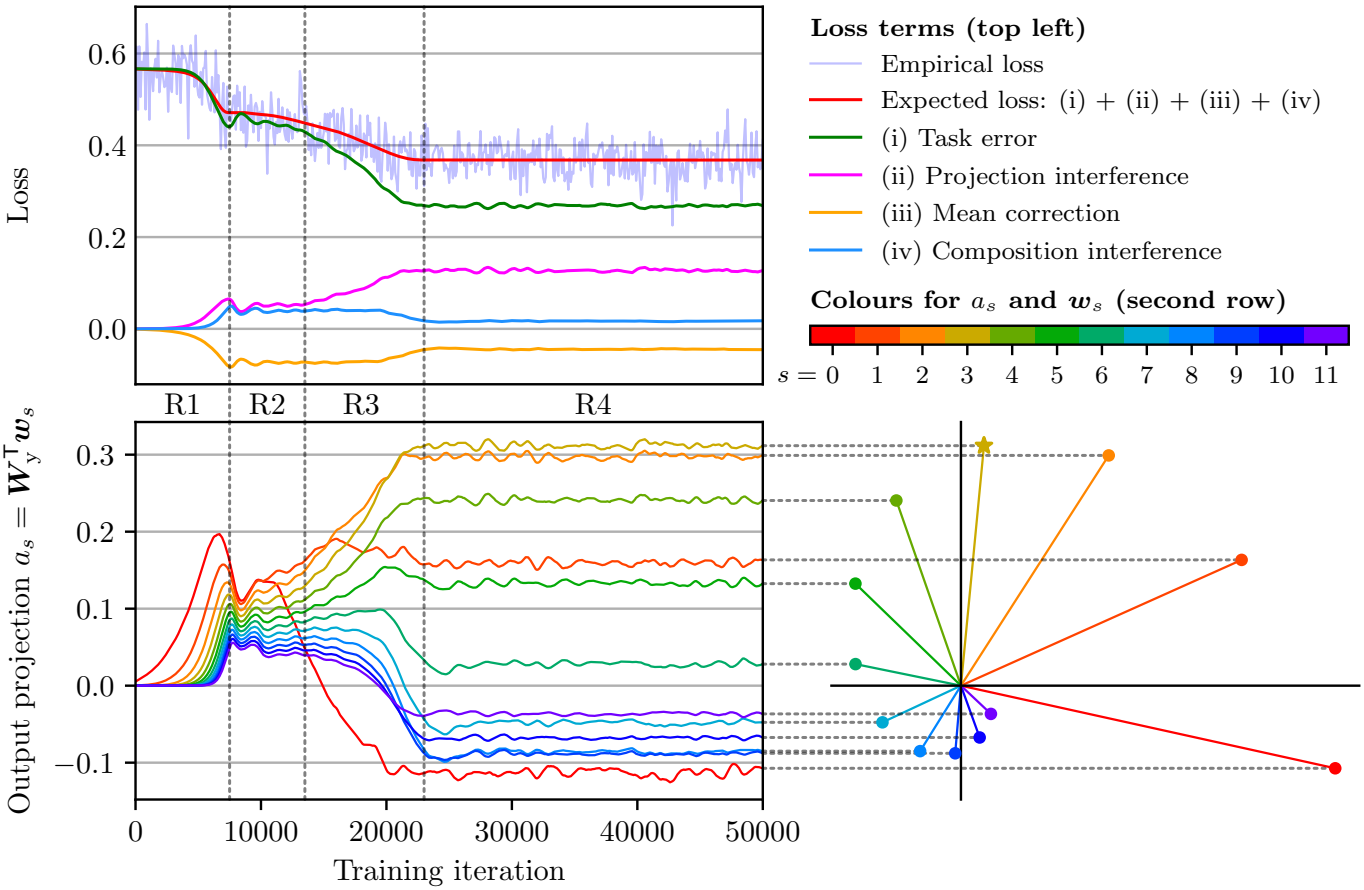


Figure 3.2: Learning dynamics of a 2-dimensional linear RNN on the 3-delay task. A linear RNN with a 2-dimensional hidden state was trained on 50,000 input sequences, each of length 20 time steps and sparsity 0.9.[1] *Top left*: the empirical training loss curve against the expected loss given by $\sum_{t=1}^{T} L^{(t)}$ and its four components. *Bottom left*: the value of each a_s (for $0 \leq s < 12$) over the course of training. *Bottom right*: the final position of the corresponding $\boldsymbol{w}_s$ vectors in the 2-dimensional hidden state, with the entire system rotated and scaled appropriately such that the y-component of each $\boldsymbol{w}_s$ is a_s. The output feature, $\boldsymbol{w}_3$, is marked with a star and, as expected, has the highest projection onto $\boldsymbol{W}_y$.

Secondly, we almost always see "spiral sink" arrangements, where $\boldsymbol{W}_h$ acts as a rotation and down-scaling, causing the $\boldsymbol{w}_s$ vectors to spiral into the origin. We will argue in section 3.3.2 that the shrinking effect is necessary for finite loss and that the combination of rotation and down-scaling – leading to a spiral sink in the case of a 2-dimensional hidden state – is optimal.

Finally, we observe that adjacent vectors in this particular figure are separated by approximately 30°, corresponding to a 12-spoke (dodecagonal) spiral. The 3-delay task has only 4 task-relevant features, so we might ask, for instance, why the vectors are not arranged into a square! It is not immediately obvious why the model should represent the four task-relevant features in an arrangement such as the 12-spoke spiral, in which their projection and composition interference is higher. Feed-forward models tend to *drop* unimportant features, not carve out more space for them [3].

3.3.2 Analysing the Spectral Radius and Eigenvalues of W_h

So far, we have analysed the model's behaviour by studying the loss incurred at each time step. In practice, however, we train models to minimise loss (such as mean squared-error) over an

[1] We set $U_t \sim \text{Uniform}[0, 1)$ – so $\mu = 0.5$ – and used the `AdamW` optimiser with `lr` $= 5 \times 10^{-3}$.

entire sequence, not just on individual time steps. In other words, we have so far used our interpretation of $L^{(t)}$ to infer the behaviour of models that actually minimise $L_T = \frac{1}{T}\sum_{t=1}^{T} L^{(t)}$.

In this section we argue that this is a reasonable perspective to take for the study of well-performing models, whose mean squared-error does not diverge to ∞. In doing so, we show that the recurrent weight matrix $\boldsymbol{W}_\text{h}$ of such models has spectral radius $\rho(\boldsymbol{W}_\text{h}) < 1$. We conclude with an empirical verification that, at least in the case of a 2-dimensional hidden state, the optimal arrangement for the k-delay task is a spiral sink, as suggested by figure 3.2.

In this work, we are mostly interested in optimal models. As such, we restrict our attention to models whose mean squared-error L_T remains finite as $T \to \infty$. Models that do not satisfy this may perform well on short sequences but will see their performance degrade over time and, for sufficiently long sequences, will always be outperformed by models that do satisfy this assumption. Therefore, we only consider models that satisfy

$$\lim_{T\to\infty} L_T = \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} L^{(t)} < \infty.$$

If $L^{(t)}$ diverges to ∞ as $t \to \infty$, then L_T also diverges to ∞ as $T \to \infty$. To avoid this, therefore, we require that $L^{(t)}$ remains finite. We now prove that this is essentially only possible if the spectral radius of $\boldsymbol{W}_\text{h}$ is $\rho(\boldsymbol{W}_\text{h}) < 1$.

Proof. Assume for the sake of contradiction that $\rho(\boldsymbol{W}_\text{h}) \geq 1$ and that $L^{(t)}$ does not diverge to infinity. An alternative form of $(*)$ is

$$L^{(t)} = \left(p\nu - p^2\mu^2\right)(a_k - 1)^2 + \left(p\nu - p^2\mu^2\right)\sum_{s\neq k}^{t-1} a_s^2 + p^2\mu^2\left(\sum_{s=0}^{t-1} a_s - 1\right)^2, \qquad (\dagger)$$

which we derive in appendix A.1. Noting that $p\nu - p^2\mu^2 = \text{Var}(X_t) > 0$, we see that each term in this sum is non-negative.[2] Hence, for $L^{(t)}$ to not diverge to infinity, the term $\sum_{s=0}^{\infty} a_s^2$ must converge, so we require $\lim_{s\to\infty} a_s = 0$. Since $\rho(\boldsymbol{W}_\text{h}) \geq 1$, however, we have

$$\lim_{s\to\infty} a_s = \lim_{s\to\infty} \boldsymbol{W}_\text{y}^\top \boldsymbol{W}_\text{h}^s \boldsymbol{W}_\text{x} = \boldsymbol{W}_\text{y}^\top \left(\lim_{s\to\infty} \boldsymbol{W}_\text{h}^s\right) \boldsymbol{W}_\text{x} = \boldsymbol{W}_\text{y}^\top \boldsymbol{W}_\text{h}^\infty \boldsymbol{W}_\text{x},$$

where we denote $\boldsymbol{W}_\text{h}^\infty = \lim_{s\to\infty} \boldsymbol{W}_\text{h}^s \neq \boldsymbol{0}$. This limit is zero precisely when

$$\boldsymbol{W}_\text{x} \in \ker(\boldsymbol{W}_\text{h}^\infty) \quad \text{or} \quad \boldsymbol{W}_\text{y} \in \ker\left(\boldsymbol{W}_\text{h}^{\infty\top}\right) \quad \text{or} \quad \boldsymbol{W}_\text{y} \perp \boldsymbol{W}_\text{h}^\infty \boldsymbol{W}_\text{x}.$$

Since $\boldsymbol{W}_\text{h}^\infty \neq 0$, we have $\text{rank}(\boldsymbol{W}_\text{h}^\infty) = \text{rank}\left(\boldsymbol{W}_\text{h}^{\infty\top}\right) > 0$. Hence, by the rank-nullity theorem, $\dim(\ker(\boldsymbol{W}_\text{h}^\infty)) = N_h - \text{rank}(\boldsymbol{W}_\text{h}^\infty) < N_h$ and $\dim\left(\ker\left(\boldsymbol{W}_\text{h}^{\infty\top}\right)\right) = N_h - \text{rank}\left(\boldsymbol{W}_\text{h}^{\infty\top}\right) < N_h$. Thus both $\ker(\boldsymbol{W}_\text{h}^\infty)$ and $\ker\left(\boldsymbol{W}_\text{h}^{\infty\top}\right)$ have Lebesgue measure zero. If $\boldsymbol{W}_\text{x} \notin \ker(\boldsymbol{W}_\text{h}^\infty)$, we require the third case, where $\boldsymbol{W}_\text{y}^\top$ must lie on the $(N_h - 1)$-dimensional hyperplane orthogonal to $\boldsymbol{W}_\text{h}^\infty \boldsymbol{W}_\text{x}$. This is again a proper subspace of $\mathbb{R}^{N_h}$ with Lebesgue measure zero. Hence, if $\rho(\boldsymbol{W}_\text{h}) \geq 1$, then the set of solutions for which $L^{(t)}$ remains finite has measure zero, meaning that $L^{(t)}$ almost surely diverges to infinity. Therefore, taking the contrapositive, if $L^{(t)}$ remains finite, then $\rho(\boldsymbol{W}_\text{h}) < 1$ almost surely. ∎

Intuitively, we have shown that if $\rho(\boldsymbol{W}_\text{h}) \geq 1$, then $L^{(t)}$ diverges to infinity except if $\boldsymbol{W}_\text{x}$, $\boldsymbol{W}_\text{h}$ and $\boldsymbol{W}_\text{y}$ *precisely* (not approximately) satisfy certain conditions. We can be confident that these conditions are not satisfied by models in practice: it would require the optimiser to balance the model parameters on an infinitely thin "knife edge", which is practically impossible in floating-point arithmetic. This means we can safely restrict our attention to models satisfying $\rho(\boldsymbol{W}_\text{h}) < 1$. Given this, we can show that $L^{(t)}$ not only avoids diverging to ∞, but converges to a particular finite value.

[2] We ignore the trivial edge case of $\text{Var}(X_t) = 0$, which corresponds to a constant-valued input sequence.

Proof. It is clear from (†) that a finite $\lim_{t\to\infty} L^{(t)}$ exists if both $\sum_{s=0}^{\infty} a_s$ and $\sum_{s=0}^{\infty} a_s^2$ converge. We show this is the case given that $\rho(\boldsymbol{W}_\mathrm{h}) < 1$, using matrix analysis results from [21, ch.5]. For any $\varepsilon > 0$, there exists some vector norm $\|\cdot\|$ such that the induced matrix norm satisfies $\|\boldsymbol{W}_\mathrm{h}\| \leq \rho(\boldsymbol{W}_\mathrm{h}) + \varepsilon$. Choose $\varepsilon < 1 - \rho(\boldsymbol{W}_\mathrm{h})$, so $\|\boldsymbol{W}_\mathrm{h}\| < 1$. Let $\|\cdot\|_*$ denote the corresponding dual norm, so $|\boldsymbol{v}^\mathsf{T}\boldsymbol{w}| \leq \|\boldsymbol{v}\|_*\|\boldsymbol{w}\|$ for all $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{R}^{N_h}$. Then

$$\sum_{s=0}^{\infty} a_s \leq \sum_{s=0}^{\infty} |a_s| = \sum_{s=0}^{\infty} |\boldsymbol{W}_\mathrm{y}^\mathsf{T}\boldsymbol{W}_\mathrm{h}^s\boldsymbol{W}_\mathrm{x}| \leq \|\boldsymbol{W}_\mathrm{y}^\mathsf{T}\|_*\|\boldsymbol{W}_\mathrm{x}\| \sum_{s=0}^{\infty} \|\boldsymbol{W}_\mathrm{h}\|^s = \frac{\|\boldsymbol{W}_\mathrm{y}^\mathsf{T}\|_*\|\boldsymbol{W}_\mathrm{x}\|}{1 - \|\boldsymbol{W}_\mathrm{h}\|} < \infty,$$

$$\sum_{s=0}^{\infty} a_s^2 = \sum_{s=0}^{\infty} |a_s|^2 = \sum_{s=0}^{\infty} |\boldsymbol{W}_\mathrm{y}^\mathsf{T}\boldsymbol{W}_\mathrm{h}^s\boldsymbol{W}_\mathrm{x}|^2 \leq \|\boldsymbol{W}_\mathrm{y}^\mathsf{T}\|_*^2\|\boldsymbol{W}_\mathrm{x}\|^2 \sum_{s=0}^{\infty} \|\boldsymbol{W}_\mathrm{h}\|^{2s} = \frac{\|\boldsymbol{W}_\mathrm{y}^\mathsf{T}\|_*^2\|\boldsymbol{W}_\mathrm{x}\|^2}{1 - \|\boldsymbol{W}_\mathrm{h}\|^2} < \infty.$$

Hence the two infinite series converge and so if $\rho(\boldsymbol{W}_\mathrm{h}) < 1$, then $\lim_{t\to\infty} L^{(t)}$ converges. ■

We have established that models satisfying $\lim_{T\to\infty} L_T < \infty$ require $\rho(\boldsymbol{W}_\mathrm{h}) < 1$ and shown that, as a result, $\lim_{t\to\infty} L^{(t)}$ converges to a particular finite value. Therefore, we can take the Cesàro mean [22] to show that the mean squared-error L_T converges to the same value:

$$\lim_{T\to\infty} L_T = \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} L^{(t)} = \lim_{t\to\infty} L^{(t)}.$$

This means that, the long-term behaviour of a model whose mean squared-error does not diverge to infinity can be understood by analysing $L^{(t)}$ for sufficiently large t.

In doing this proof, we have seen that $\rho(\boldsymbol{W}_\mathrm{h}) < 1$. This is known as the echo state property and can be interpreted as the model forgetting old inputs over time. This constraint guarantees the shrinking aspect of a spiral sink. We now verify that the spiral behaviour, specifically, is the optimal solution in 2 dimensions. To do this, we train linear RNNs parameterised such that the trace and determinant of $\boldsymbol{W}_\mathrm{h} \in \mathbb{R}^{2\times 2}$ is fixed. Specifically, for each desired trace-determinant pair (τ, δ), we optimise over the 2-dimensional manifold

$$\mathcal{M} = \left\{ \boldsymbol{W}_\mathrm{h} \in \mathbb{R}^{2\times 2} : \mathrm{tr}(\boldsymbol{W}_\mathrm{h}) = \tau, \det(\boldsymbol{W}_\mathrm{h}) = \delta \right\},$$

which we parameterise by $(\theta_1, \theta_2) \in \mathbb{R}^2$ using the map

$$\varphi(\theta_1, \theta_2) = \begin{bmatrix} \theta_1 & \frac{\theta_1(\tau - \theta_1) - \delta}{\exp(\theta_2)} \\ \exp(\theta_2) & \tau - \theta_1 \end{bmatrix},$$

where θ_2 is exponentiated to ensure φ is bijective. We verify that

$$\mathrm{tr}(\varphi(\theta_1, \theta_2)) = \theta_1 + \tau - \theta_1 = \tau,$$
$$\det(\varphi(\theta_1, \theta_2)) = \theta_1(\tau - \theta_1) - \frac{\theta_1(\tau - \theta_1)}{\exp(\theta_2)} \exp(\theta_2) = \delta.$$

We sweep through a grid of points in the square $(\tau, \delta) \in [-2, 2]^2$ and for each point, we train a linear RNN parameterised as above. The loss achieved by each model under various task conditions is shown in figure 3.4 on the following page. Each plot can be thought of as a trace-determinant slice of the k-delay loss landscape.

First of all, we note that in every case, the optimal solution (brightest point on the plot) is found in the region that corresponds to spiral sinks. This provides strong empirical evidence that the globally optimal solution must be a spiral sink. This makes intuitive sense: rotation is used to implement an approximate delay-line, while the gradual shrinking of vectors facilitates the forgetting of old inputs.

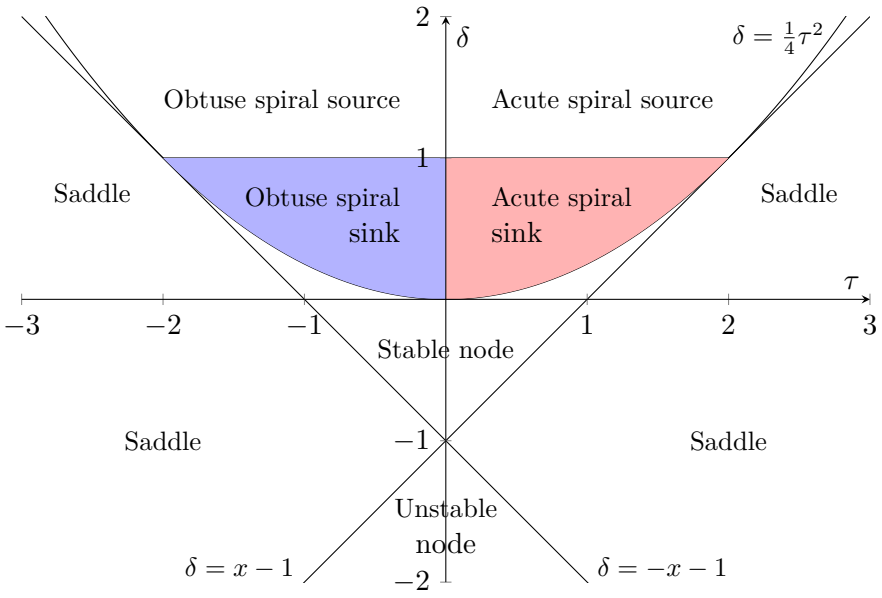


Figure 3.3: Trace-determinant classification of 2-dimensional discrete linear dynamical systems, adapted from [23]. Stable systems occupy the triangular region enclosed by the lines $\delta = 1$, $\delta = x - 1$ and $\delta = -x - 1$. Within this triangle, spiral sinks are found above the parabola $\delta = \frac{1}{4}\tau^2$.

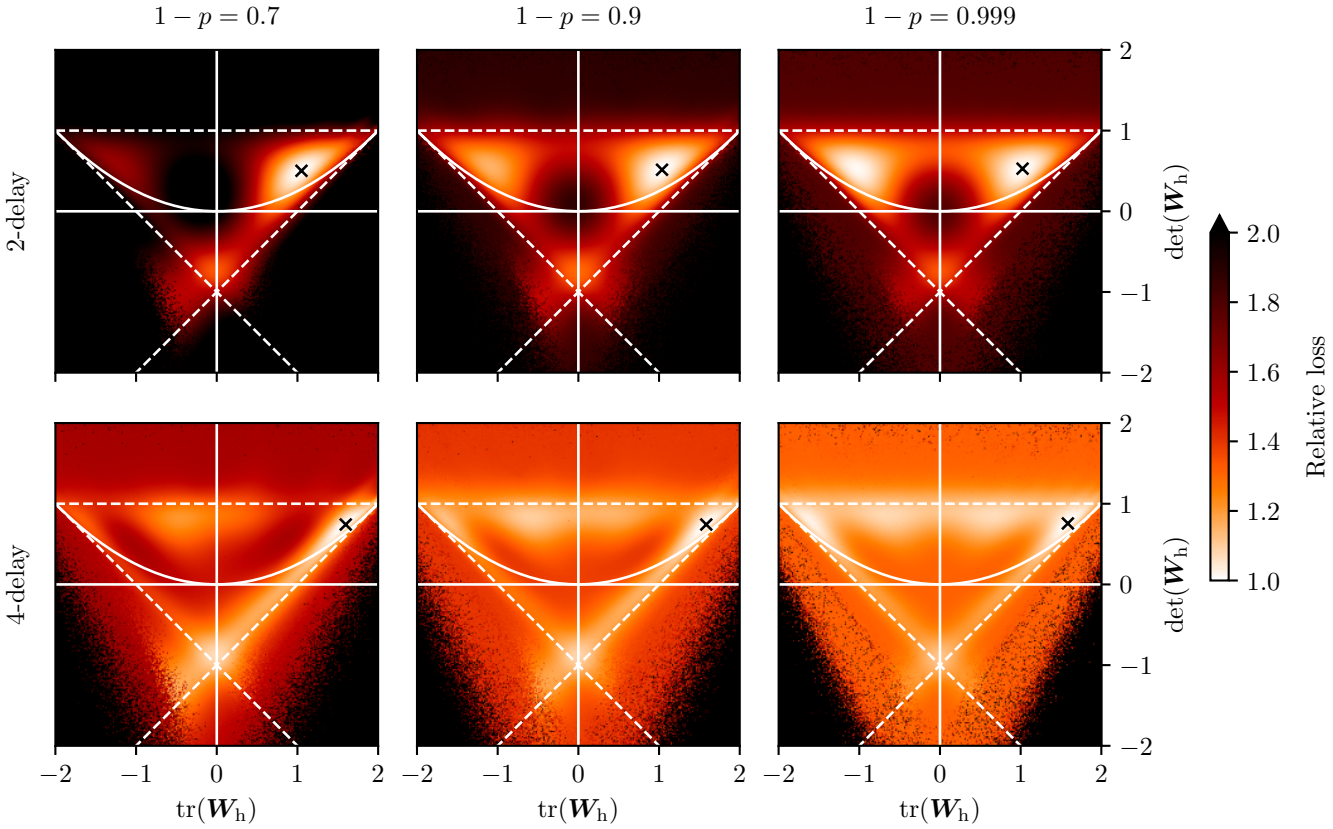


Figure 3.4: Loss landscape of the 2-delay and 4-delay tasks at 3 sparsity levels for linear RNNs with 2-dimensional $\boldsymbol{W}_\text{h}$, shown in terms of $\text{tr}(\boldsymbol{W}_\text{h})$ and $\det(\boldsymbol{W}_\text{h})$. At each point on the trace-determinant plane a linear model with 2-dimensional hidden state was parameterised, as described above, with a fixed trace and determinant. Each model was trained on 1000 sequences of length 20. The final training loss is displayed as a multiple of the lowest training loss achieved by any of the models. The best-performing model for each task is marked by a cross. Standard lines and curves used to classify discrete dynamical systems are overlaid in white; refer to figure 3.3 for interpretation.

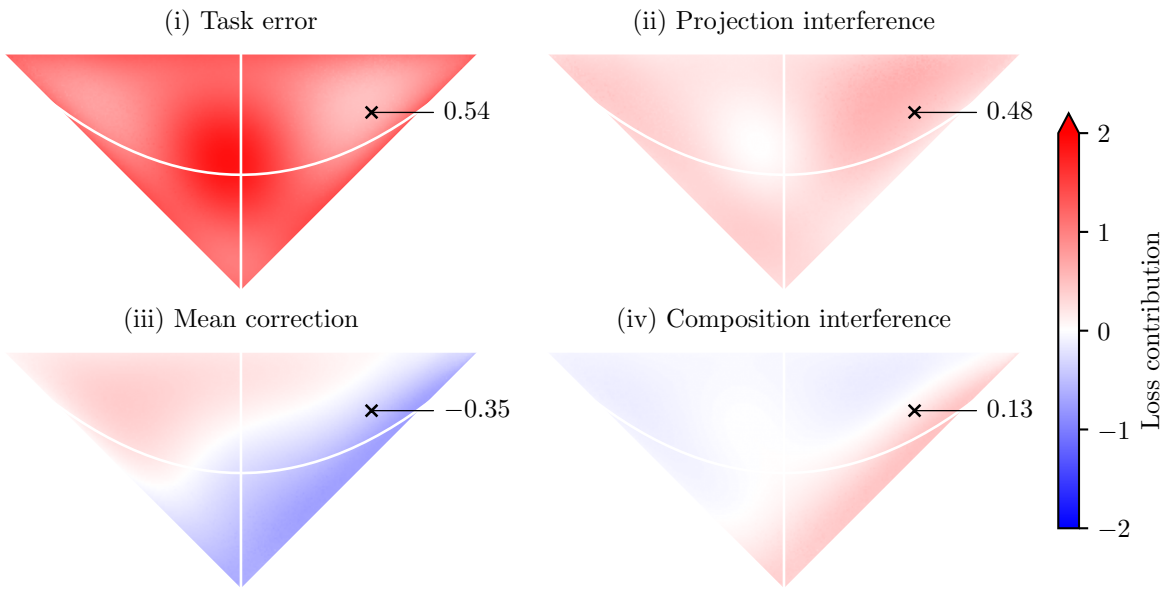


Figure 3.5: Trace-determinant loss landscape of the 2-delay task at 0.7 sparsity for stable linear RNNs with 2-dimensional $\boldsymbol{W}_{\text{h}}$, decomposed into interpretable terms. The best-performing model (in terms of overall empirical loss) is marked with a cross and annotated with its value for each of the four loss terms.

Interestingly, the plots become increasingly symmetric with increasing sparsity. In the low-sparsity case for both the 2-delay and 4-delay tasks, the global optimum clearly has $\text{tr}(\boldsymbol{W}_{\text{h}}) > 0$, which implies an acute rotation angle. This can be seen by writing the eigenvalues of $\boldsymbol{W}_{\text{h}}$ as

$$\lambda_{1,2} = \frac{t}{2} \pm \frac{i}{2}\sqrt{4\delta - \tau^2}$$

and observing that $t < 0$ corresponds to $\arg(\lambda_{1,2}) \in \left(\frac{\pi}{2}, \frac{3\pi}{2}\right)$ (obtuse rotation angle) while $t > 0$ corresponds to $\arg(\lambda_{1,2}) \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$ (acute rotation angle).

The asymmetry is most clearly visible in the figure for the 2-delay task at 0.7 sparsity. In figure 3.5, we decompose this particular loss landscape to investigate the reason for acute-angled rotation being preferred. While none of the terms are perfectly symmetric in $\text{tr}(\boldsymbol{W}_{\text{h}})$, it is clear that the mean correction term is largely driving this behaviour: for acute spirals, it can reduce loss, while for most instances of obtuse spirals, it increases the loss. Unsurprisingly, there exists a trade-off between this term and the others (in particular, composition interference largely seems positive where mean correction is negative, and vice versa), but evidently the optimal balance is firmly in the acute spiral region. Overall, there seems to be a region in which task error, mean correction and composition interference are all relatively low, while projection interference is relatively high – this corresponds precisely with the lowest-loss region in figure 3.4 and is exactly the sacrifice we saw the model make in figure 3.2.

3.3.3 Exploring Mean Correction and Bias

The mean correction term is perhaps the least intuitive component of the loss. Fundamentally, it arises from the fact that inputs carry not only short-term information about how the output should vary in the next few time steps, but also task-wide information about what the mean of the input distribution is. In particular, even task-irrelevant temporal features, which cannot ever contribute to reducing task error, still carry useful information about the mean of the data. Theoretically, then, an actual bias term in the output should fulfil the same purpose. In this section, we consider the effect of using an explicit bias term on the loss and show analytically

that in the limit of infinite sequence length, using a bias in the output recovers the simplified loss expression (**) with no mean correction or composition inteference terms.

To begin, we repeat the loss decomposition for an RNN with linear recurrence that includes an explicit bias term $b \in \mathbb{R}$ in its output:

$$\boldsymbol{h}_0 = \boldsymbol{0}, \qquad \boldsymbol{h}_t = \boldsymbol{W}_{\mathrm{x}} x_t + \boldsymbol{W}_{\mathrm{h}} \boldsymbol{h}_{t-1}, \qquad y_t = \boldsymbol{W}_{\mathrm{y}}^{\top} \boldsymbol{h}_t + b.$$

The expected loss incurred at time t is now

$$\begin{aligned} L^{(t)} &= \mathbb{E}\left[\left(X_{t-k} - \sum_{s=0}^{t-1} a_s X_{t-s} - b\right)^2\right] \\ &= \ldots - 2b\,\mathbb{E}\left[X_{t-k}\right] + 2b \sum_{s=0}^{t-1} a_s\, \mathbb{E}[X_{t-s}] + b^2 \\ &= \ldots - 2bp\mu + 2bp\mu \sum_{s=0}^{t-1} a_s + b^2, \end{aligned}$$

where we omit terms that remain the same as before. First, we remark that when $\mu = 0$, the only new term that remains is b^2, so the optimal bias is $b = 0$, as expected. In fact, we can analytically compute the optimal bias in the general case. To do this, we recognise that the model parameters are optimised against the loss incurred over an entire input sequence, not just one time step. If the model is trained on sequences of length $T \geq 1$, this total loss is given by

$$L_T = \sum_{t=1}^{T} L^{(t)}.$$

Taking partial derivatives with respect to b, we find

$$\frac{\partial L_T}{\partial b} = \sum_{t=1}^{T} \frac{\partial L^{(t)}}{\partial b} = 2p\mu \sum_{t=1}^{T} \left(\sum_{s=0}^{t-1} a_s - 1\right) + 2bT \quad \text{and} \quad \frac{\partial^2 L_T}{\partial b^2} = 2T > 0.$$

The second partial derivative is a positive constant, so L_T is a strongly convex quadratic in b. Consequently, there exists a unique value of b that minimises L_T, given by

$$b_T^* = p\mu - \frac{p\mu}{T} \sum_{t=1}^{T} \sum_{s=0}^{t-1} a_s = p\mu - \frac{p\mu}{T} \sum_{s=0}^{T-1} (T-s) a_s = p\mu \left(1 - \sum_{s=0}^{T-1} a_s + \frac{1}{T} \sum_{s=0}^{T-1} s a_s\right).$$

We now consider the case of an infinitely long input sequence. As shown in section 3.3.2, the loss at time t, $L^{(t)}$, diverges almost surely if the spectral radius of the recurrent weight matrix $\rho(\boldsymbol{W}_{\mathrm{h}}) \geq 1$. We are interested in globally optimal solutions to the task, so we assume the more reasonable case of $\rho(\boldsymbol{W}_{\mathrm{h}}) < 1$. Then, for some constant $c \in [0, \infty)$, we have

$$\left| \sum_{s=0}^{\infty} s a_s \right| \leq \sum_{s=0}^{\infty} s |a_s| \leq c \sum_{s=0}^{\infty} s \rho^s = \frac{c\rho}{(1-\rho)^2} < \infty.$$

Hence, in the limit of $T \to \infty$, the optimal bias is

$$b_\infty^* = p\mu \left(1 - \sum_{s=0}^{\infty} a_s\right).$$

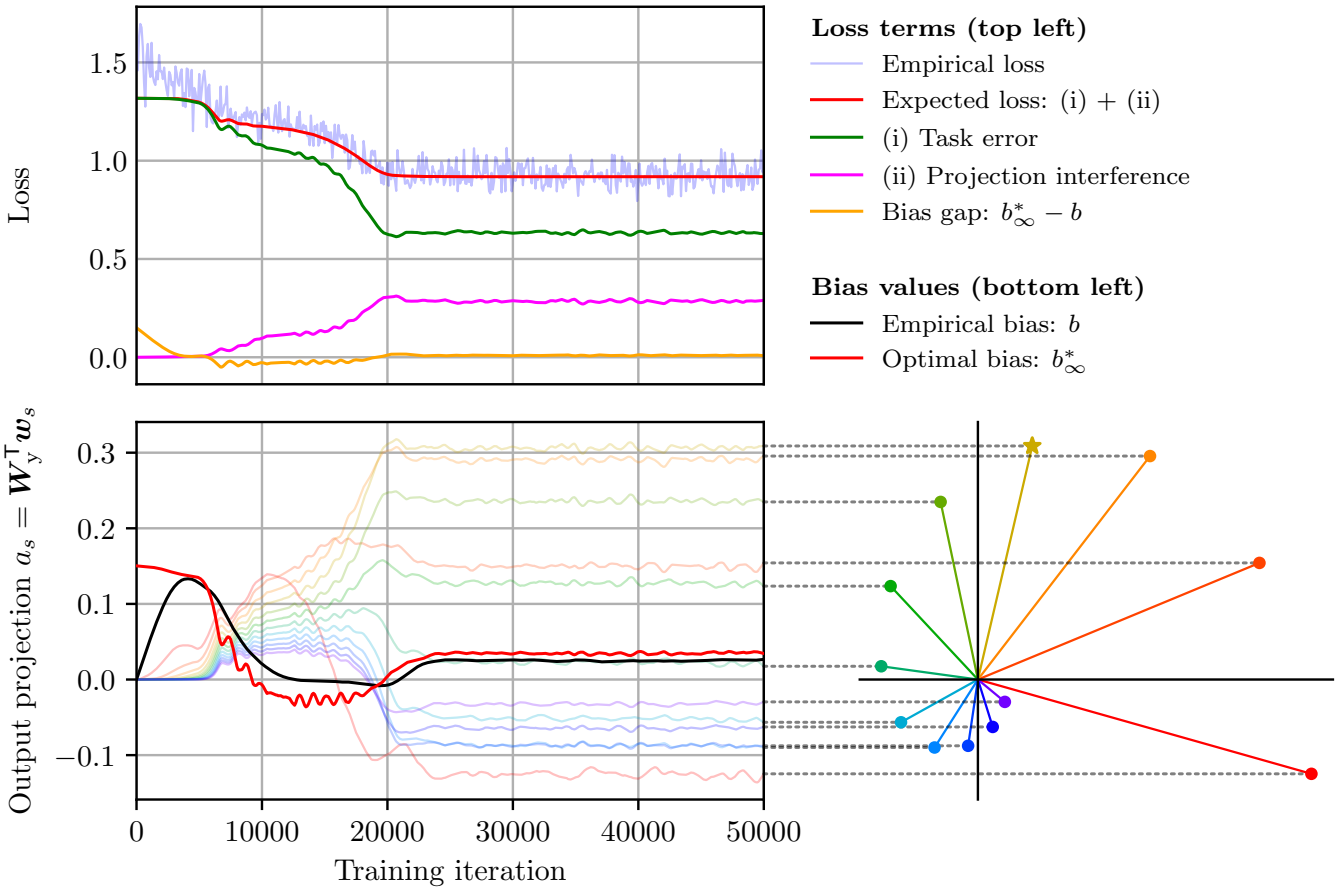


Figure 3.6: Learning dynamics of a 2-dimensional linear RNN with bias on the 3-delay task. A linear RNN with a 2-dimensional hidden state and a bias term in its output was trained on 50,000 input sequences, each of length 20 time steps and sparsity 0.7.[3] *Top left*: the expected loss is now approximated using $\lim_{t\to\infty} L^{(t)}$ assuming $b = b^*_\infty$, without a mean correction or projection interference component. The gap between the observed bias b and optimal b^*_∞ is plotted. *Bottom left*: the values of b and b^*_∞.

We now substitute b^*_∞ back into the expression for $L^{(t)}$. This allows us to focus on task performance in terms of the recurrence, assuming an optimal bias. Taking $t \to \infty$, we find

$$\begin{aligned}
\lim_{t\to\infty} L^{(t)} &= \ldots - 2p^2\mu^2\left(1 - \sum_{s=0}^{\infty} a_s\right) + 2p^2\mu^2\left(1 - \sum_{s=0}^{\infty} a_s\right)\sum_{s=0}^{\infty} a_s + p^2\mu^2\left(1 - \sum_{s=0}^{\infty} a_s\right)^2 \\
&= \ldots - p^2\mu^2 + 2p^2\mu^2\sum_{s=0}^{\infty} a_s - p^2\mu^2\left(\sum_{s=0}^{\infty} a_s\right)^2 \\
&= \left(p\nu - p^2\mu^2\right) - 2\left(p\nu - p^2\mu^2\right)a_k + \left(p\nu - p^2\mu^2\right)\sum_{s=0}^{\infty} a_s^2 \\
&\propto (a_k - 1)^2 + \sum_{s\neq k}^{\infty} a_s^2.
\end{aligned}$$

Remarkably, this recovers the simple form of the loss (**). This implies that, with an explicit bias term, a global optimiser of the k-delay task with non-zero input mean behaves identically to the global optimiser of the zero-mean case in the limit of infinite sequence length.

[3] Again, we set $U_t \sim \text{Uniform}[0, 1)$ and used the `AdamW` optimiser with `lr` $= 5 \times 10^{-3}$.

This is verified empirically in figure 3.6 on the preceding page. Here we see that even when training on sequences as short as 20 time steps, the model learns a bias very close to our predicted b^*_∞. Notably, although the expected loss now only accounts for task error and projection interference, it still closely follows the empirical loss, meaning that the simple form of the loss has been recovered. The input data has low sparsity and non-zero mean, so this effect can only have arisen due to the bias term, thus validating our theoretical finding.

There is, of course, some difference between the empirical and expected loss, where the "bias gap" is also large. The model does not learn the optimal bias immediately and, in fact, a non-zero gap remains throughout training. This is not entirely surprising: our analytical approach does not model actual optimisation and essentially assumes that, for any current values of $\boldsymbol{W}_\text{x}$, $\boldsymbol{W}_\text{h}$ and $\boldsymbol{W}_\text{y}$, the optimal bias b^*_∞ is immediately in use. In practice, on each parameter update, the optimiser does not adjust b according to the post-update values of other parameters, but according to the pre-update values; so the new b is immediately outdated. Furthermore, the iterative nature of gradient descent means that b approaches b^*_∞ in steps, but b^*_∞ itself will change in that time. Essentially, b is both reacting to outdated information and chasing a moving target. This effect can be seen in figure 3.6: the b^*_∞ curve tends to "move first" and there is some delay before the actual parameter b catches up. The remaining discrepancy (such as between the final values of b and b^*_∞) is due to using short sequences of just 20 time steps and disappears as the sequence length is increased.

3.3.4 Understanding Composition Interference

We have now seen three ways in which the effect of composition interference can be made negligible in the k-delay task. Each works by recovering the simple form of the loss, where the model can behave as if the data has mean zero:

1. If the mean of the input data is zero, so $\mu = 0$, then any composition interference that occurs to increase the loss is, by symmetry, equally likely to occur in the opposite direction to decrease the loss, so the average effect of composition interference on the loss is zero.

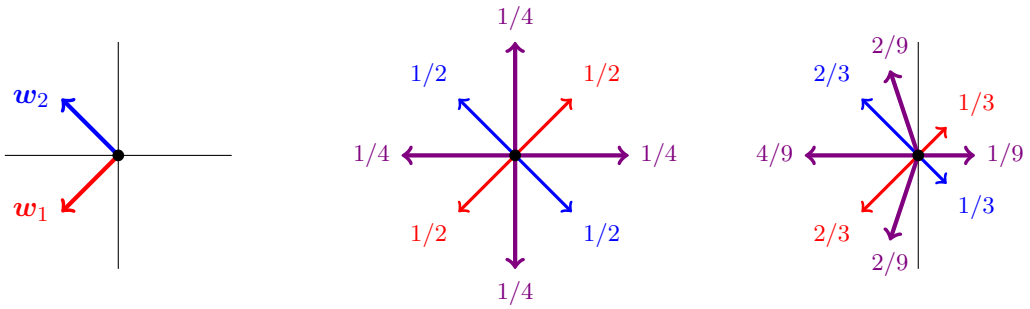
2. If the input is extremely sparse, then the effective input mean is close to zero, so the impact of the non-zero mean is suppressed and we essentially recover the case above. As an alternative explanation, in the extremely sparse regime, it is very unlikely for multiple features to activate simultaneously, so composition interference becomes a negligible factor.

3. If the model has a bias term in its output, this alone can learn to offset the non-zero mean, allowing the remaining parameters of the model to focus solely on learning the variance, thus behaving as if the input data had mean zero.

Unintuitively, composition interference seems closely linked to the mean of the input data. This is because the overall effect of composition interference is to amplify existing bias in the data, as visualised in figure 3.7 on the next page. If the input is unbiased, then, on average, composition interference completely cancels itself out.

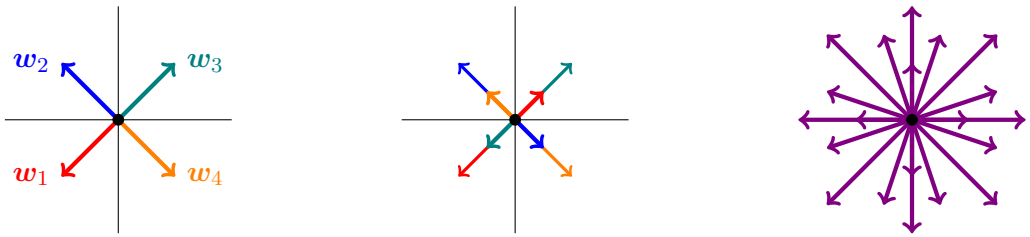
This perspective also elucidates why we often see antipodal feature pairs in trained models. As depicted in figure 3.8 on the following page, for i.i.d. input data and vectors of equal norm, arranging the vectors in antipodal pairs makes the set of sparse activations – all possible activations where only one feature is active – symmetric. This is precisely the property that did not hold in the non-antipodal case. Since composition interference, by definition, composes new activations from this set, we can deduce that the set of all activations resulting from composition interference must also be symmetric. As a result, the composition is not biased in any particular direction and so, as before, it entirely cancels itself out.

Note that we have considered a simplified scenario here, but the intuition should extend to cases where the input data is not i.i.d. or the features are not represented by vectors of equal



(a) Non-antipodal pair **(b)** Zero mean input **(c)** Non-zero mean input

Figure 3.7: Input data with non-zero mean causes an asymmetric activations for non-antipodal pairs of features. In (a), we introduce two features orthogonally represented by $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ of equal norm. As seen in (b), if the input data has zero mean, then each input is positive with probability 1/2 and negative with probability 1/2, so the resulting activations are equally likely to be composed of the positive directions $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ as they are to be composed of the negative directions $-\boldsymbol{w}_1$ and $-\boldsymbol{w}_2$, as shown by the vectors in purple. These sum to zero, so the average effect of composition interference is null. In (c), the input data has non-zero mean and is instead positive with probability 2/3. This skews all activations in the direction of $\boldsymbol{w}_1 + \boldsymbol{w}_2$. Since $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ are non-antipodal, $\boldsymbol{w}_1 + \boldsymbol{w}_2 \neq \boldsymbol{0}$, so there is a direction in activation space that composition interference is biased towards constructing.



(a) Product of antipodal pairs **(b)** Features activate symmetrically **(c)** Composition is symmetric

Figure 3.8: A tegum product of antipodal pairs (a) can achieve zero composition interference even when the input data has non-zero mean. The situation here is the same as in figure 3.7(c) but with two more features $\boldsymbol{w}_3$ and $\boldsymbol{w}_4$ arranged carefully to restore symmetry: the set of all possible sparse activations is symmetric (b), so any composition of those activations remains symmetric (c). Thus, for every composition of activations that increases the loss, there exists an equal and opposite composition of activations that decreases it; hence, again, the overall effect of composition interference is zero.

norm. In such cases, antipodal pairs may no longer be best, but some other arrangement of features may lead to a symmetric set of sparse activations.

The symmetry argument does, however, rely on the assumption that opposite directions in activation space correspond to opposite outputs. This is true in the entirely linear RNNs that we have discussed so far, because $y_t(-\boldsymbol{h}_t) = -\boldsymbol{W}_{\text{y}}^{\top}\boldsymbol{h}_t = -y_t(\boldsymbol{h}_t)$. More generally, it is true for any RNN that reads its hidden state out with an *odd* activation function σ that satisfies $\sigma(-x) = -\sigma(x)$. In other cases, such as that of the ReLU activation function (which we consider in the following section), the only situation in which composition interference is negligible is in the extremely sparse regime, where the simultaneous activation of features is vanishingly rare.

Chapter 4

Temporal Superposition in Non-Linear RNNs

This chapter extends our analysis to RNNs that include non-linearities. We begin with an intermediate case: a model that has linear recurrence but non-linear read-out. We derive an approximation to the loss $L^{(t)}_{\text{ReLU}}$ under the assumption of the echo state property and high input sparsity. Interpreting the result geometrically, we discover that, there exists a half-space in activation space that becomes completely free of all interference in the extremely sparse regime ($p \to 0$). We verify empirically that models learn to exploit this interference-free half-space in cases of high sparsity, matching our theoretical prediction.

We observe a phase transition in the optimal geometric arrangement for these models as sparsity is increased. Interestingly, we see evidence of an unstable regime in between the dense and sparse regimes and a corresponding metastable state that makes the internal representation of the model unpredictable within this unstable regime.

We then consider an RNN with non-linear recurrence. We carefully construct a non-linear recurrence for which the linear representation hypothesis is provably true in the limit of perfect sparsity. The result is a similar geometric interpretation to the case of linear recurrence but with increased model expressivity; we verify empirically that the internal representations learnt by these models under high sparsity is almost perfectly predicted by our theoretical framework.

4.1 Analysing a Linear Model with Non-Linear Read-Out

4.1.1 Introducing a ReLU Non-Linearity in the Output

We now consider an RNN that produces its output through a ReLU activation function:

$$\boldsymbol{h}_0 = \boldsymbol{0}, \qquad \boldsymbol{h}_t = \boldsymbol{W}_{\mathrm{x}} x_t + \boldsymbol{W}_{\mathrm{h}} \boldsymbol{h}_{t-1}, \qquad y_t = \mathrm{ReLU}\left(\boldsymbol{W}_{\mathrm{y}}^{\mathsf{T}} \boldsymbol{h}_t\right).$$

Such a model can only produce non-negative outputs, so for the k-delay task, it is sensible to restrict the input distribution to be non-negative as well (as we desire $y_t = x_{t-k}$). Recall that the input distribution is $X_t = B_t U_t$, where $B_t \sim \mathrm{Bernoulli}(p)$ and U_t is are identically distributed according to any distribution. Then, since $B_t \geq 0$, we require $U_t \geq 0$ to ensure $X_t \geq 0$. For example, the distribution used by [3], where $U_t \sim \mathrm{Uniform}[0, 1]$, would satisfy this requirement.

We now prove some key results about the expected value of the ReLU function applied to functions of a random variable $Z \geq 0$. Firstly, we note that

$$\mathbb{E}[\mathrm{ReLU}(Z)] = \mathbb{E}[Z].$$

We now consider $\mathbb{E}[\mathrm{ReLU}(AZ)]$ for an arbitrary random variable $A \in \mathbb{R}$, which is permitted to be negative. Since $Z \geq 0$, we have

$$\mathbb{E}[\mathrm{ReLU}(AZ)] = \mathbb{E}[\mathrm{ReLU}(A)Z] = \mathbb{E}[\mathrm{ReLU}(A)]\,\mathbb{E}[Z].$$

In the special case of a constant $A = a$, this simplifies to $\mathbb{E}[\mathrm{ReLU}(aZ)] = \mathrm{ReLU}(a)\,\mathbb{E}[Z]$.

Now, recalling that $B_t, U_t, X_t \geq 0$ with $\mathbb{E}[B_t] = p$, $\mathbb{E}[U_t] = \mu$ and $\mathbb{E}[X_t] = p\mu$, we apply these results to the expected model output given by

$$\mathbb{E}[Y_t] = \mathbb{E}\left[\mathrm{ReLU}\left(\sum_{s=0}^{t-1} a_s X_{t-s}\right)\right].$$

Previously, we were able to apply linearity of expectation to split the expectation of a sum into a sum of expectations, but the ReLU non-linearity precludes using the same approach here. The distribution of the interior weighted sum of uniformly distributed variables is known [24], but its complexity explodes with increasing t. We will therefore opt for an approximation.

4.1.2 An Approximation of $L^{(t)}_{\mathbf{ReLU}}$ Under High Sparsity

Let ρ be the spectral radius of $\boldsymbol{W}_{\mathrm{h}}$. Then the temporal feature x_{t-s} is represented by the vector $\boldsymbol{w}_s = \boldsymbol{W}_{\mathrm{h}}^s \boldsymbol{W}_{\mathrm{x}}$ in the hidden state $\boldsymbol{h}_t$. For old features, corresponding to large s, we have $\|\boldsymbol{W}_{\mathrm{h}}^s\| \approx \rho^s$ by Gelfand's formula [21, ch.5]. As argued in section 3.3.2, we are only concerned with the case of $\rho < 1$, so for any $\varepsilon > 0$, there exists a "memory window" of length $T_\varepsilon = \lceil \log(\varepsilon)/\log(\rho) \rceil$ such that for $s \geq T_\varepsilon$, $\|\boldsymbol{W}_{\mathrm{h}}^s\| \approx \rho^s \leq \rho^{T_\varepsilon} \leq \varepsilon$. Hence the contribution of any input older than T_ε has magnitude of order $\mathcal{O}(\varepsilon)$. Intuitively, this means that if we set ε small enough (and thus T_ε large enough), we can ignore inputs older than T_ε time steps.

Therefore, the only inputs that can have a significant effect on the model's behaviour are those which arrived in the last T_ε time steps. Since each input is masked by a Bernoulli random variable, the number of non-zero inputs that arrive in T_ε time steps is distributed according to $N_\varepsilon \sim \mathrm{Binomial}(T_\varepsilon, p)$. This quantity essentially counts the number of inputs actually "in play", meaning that their effect on the hidden state has magnitude larger than ε. Hence, the probability that there are two or more such inputs is given by

$$\Pr[N_\varepsilon \geq 2] = 1 - (1-p)^{T_\varepsilon} - pT_\varepsilon\,(1-p)^{T_\varepsilon - 1} \approx \frac{p^2}{2} T_\varepsilon (T_\varepsilon - 1),$$

where the binomial approximation holds for small p (full proof in appendix A.2). Therefore, if we are willing to ignore cases that arise with probability less than some $\delta > 0$, we can approximate

the behaviour of an RNN by its behaviour on input sequences with only one non-zero input for $\Pr[N_\varepsilon \geq 2] < \delta$. This occurs when sparsity is high enough to make it vanishingly rare for two or more inputs to be "in play" simultaneously. Specifically, the approximation is valid when

$$p < \sqrt{\frac{2\delta}{T_\varepsilon(T_\varepsilon - 1)}} < \frac{\sqrt{2\delta}}{\log(1/\varepsilon)} \log(1/\rho),$$

or, equivalently,

$$\rho < \exp\left(-\frac{p \log(1/\varepsilon)}{\sqrt{2\delta}}\right) = \exp\left(\frac{p \log(\varepsilon)}{\sqrt{2\delta}}\right) = \varepsilon^{p/\sqrt{2\delta}}.$$

In particular, for arbitrarily tight $\delta, \varepsilon > 0$, there always exists a sparsity level p small enough to make the approximation valid for any given model with $\rho < 1$. Under this approximation, our analysis of the ReLU-gated model becomes tractable, as we can ignore all cases that involve two or more non-zero inputs. For instance, the expected model output becomes

$$\mathbb{E}\left[\mathrm{ReLU}\left(\sum_{s=0}^{t-1} a_s X_{t-s}\right)\right] \approx \sum_{s=0}^{t-1} \mathbb{E}\left[\mathrm{ReLU}\left(a_s X_{t-s}\right)\right] = \sum_{s=0}^{t-1} \mathbb{E}[X_{t-s}]\,\mathrm{ReLU}\left(a_s\right) = p\mu \sum_{s=0}^{t-1} \mathrm{ReLU}(a_s).$$

We now follow the steps of section 3.2.3, applying this assumption to derive an interpretable expression for the loss incurred at time t. We begin with

$$L_{\mathrm{ReLU}}^{(t)} = \mathbb{E}\left[X_{t-k}^2\right] - 2\,\mathbb{E}\left[X_{t-k}\,\mathrm{ReLU}\left(\sum_{s=0}^{t-1} a_s X_{t-s}\right)\right] + \mathbb{E}\left[\mathrm{ReLU}\left(\sum_{s=0}^{t-1} a_s X_{t-s}\right)^2\right]$$

As before, the first term is $\mathbb{E}\left[X_{t-k}^2\right] = p\nu$. To evaluate the second term, we again use the assumption that no more than one of the inputs is non-zero. There are two cases: either X_{t-k} is zero, in which case the entire term collapses to zero, or X_{t-k} is non-zero, in which case all other X_{t-s} are zero for $s \neq k$. Hence the second expectation simplifies to

$$\begin{aligned}
\mathbb{E}\left[X_{t-k}\,\mathrm{ReLU}\left(\sum_{s=0}^{t-1} a_s X_{t-s}\right)\right] &\approx \mathbb{E}\left[X_{t-k}\,\mathrm{ReLU}(a_k X_{t-k})\right] \\
&= \mathbb{E}\left[X_{t-k}^2\right]\mathrm{ReLU}(a_k) \\
&= p\nu\,\mathrm{ReLU}(a_k).
\end{aligned}$$

Similarly, in the third expectation, the only non-zero summands are those on the "diagonal" (all off-diagonal terms require two inputs to be non-zero, so we ignore them):

$$\begin{aligned}
\mathbb{E}\left[\mathrm{ReLU}\left(\sum_{s=0}^{t} a_s X_{t-s}\right)^2\right] &\approx \sum_{s=0}^{t} \mathbb{E}\left[\mathrm{ReLU}\left(a_s X_{t-s}\right)^2\right] \\
&= \sum_{s=0}^{t} \mathbb{E}\left[X_{t-s}^2\right]\mathrm{ReLU}(a_s)^2 \\
&= p\nu \sum_{s=0}^{t} \mathrm{ReLU}(a_s)^2.
\end{aligned}$$

Putting these together:

$$L_{\mathrm{ReLU}}^{(t)} = \mathbb{E}\left[X_{t-k}^2\right] - 2\,\mathbb{E}\left[X_{t-k}\,\mathrm{ReLU}\left(\sum_{s=0}^{t} a_s X_{t-s}\right)\right] + \mathbb{E}\left[\mathrm{ReLU}\left(\sum_{s=0}^{t} a_s X_{t-s}\right)^2\right]$$

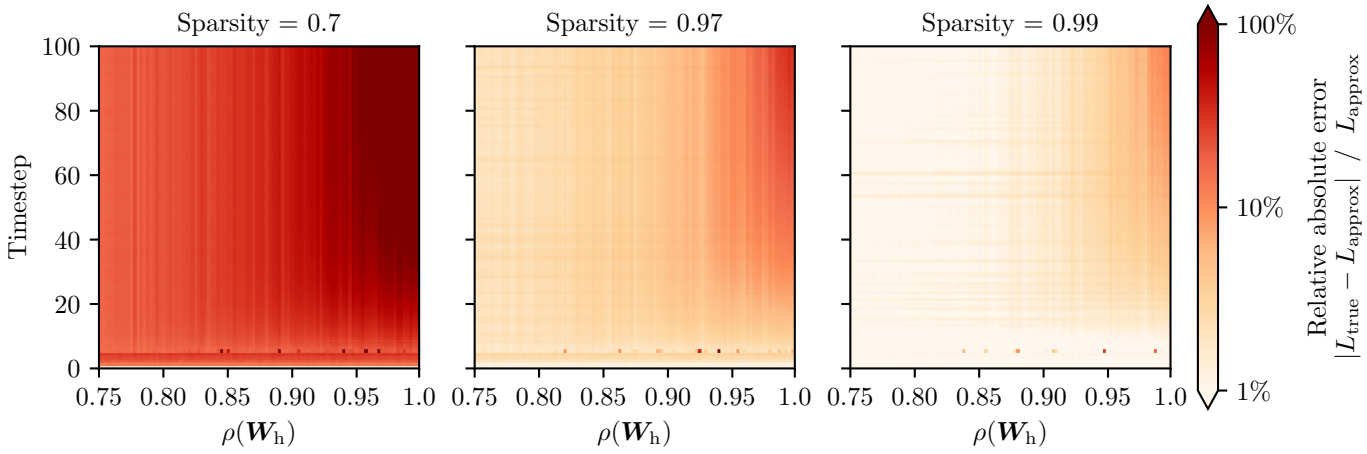


Figure 4.1: Relative error in approximating $L^{(t)}_\mathrm{ReLU}$ for high ρ. At each value of ρ_target, 500 recurrent weight matrices $\boldsymbol{W}_\mathrm{h} \in \mathbb{R}^{2\times 2}$ were randomly sampled and then linearly scaled by $\rho_\mathrm{target}/\rho(\boldsymbol{W}_\mathrm{h})$, thus setting the spectral radius of each matrix to ρ_target. Each $\boldsymbol{W}_\mathrm{h}$ was paired with different randomly sampled vectors $\boldsymbol{W}_\mathrm{x}, \boldsymbol{W}_\mathrm{y} \in \mathbb{R}^2$ to form a 2-dimensional linear model with a ReLU non-linearity in the output. Each random model's performance on the 5-delay task was evaluated over 100,000 sequences for each sparsity level. At each time step, $L^{(t)}_\mathrm{ReLU}$ was calculated according to the approximation and compared to the empirically observed loss. The three plots show, for each sparsity level, the relative approximation error at each time step, averaged across the 500 random models for each ρ.

$$\approx p\nu - p\nu\,\mathrm{ReLU}(a_k) + p\nu \sum_{s=0}^{t} \mathrm{ReLU}(a_s)^2$$

$$\propto \big(\mathrm{ReLU}(a_k) - 1\big)^2 + \sum_{s\neq k}^{t} \mathrm{ReLU}(a_s)^2.$$

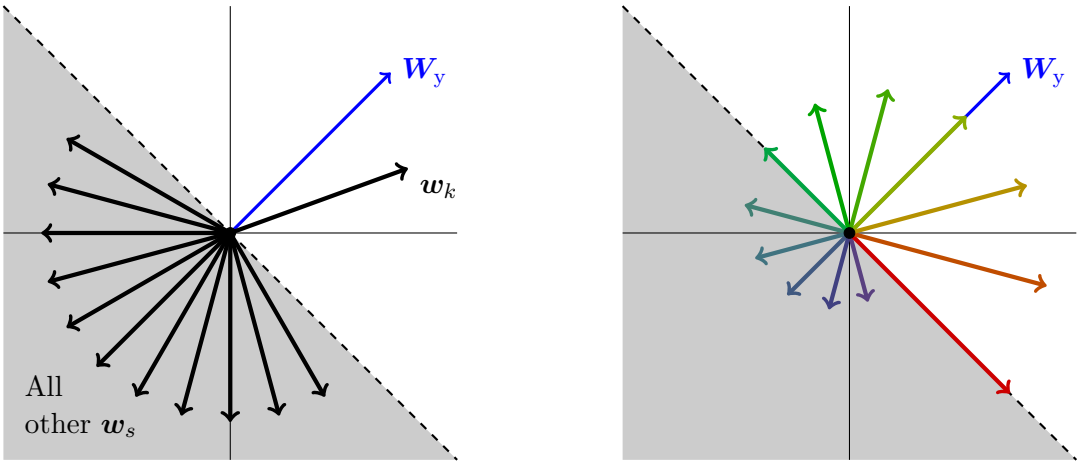
See figure 4.1 for an empirical validation of this approximation. In the dense regime, the approximation is generally inaccurate, with over 10% approximation error for all $\rho \geq 0.75$. In cases of high sparsity ($p < 0.1$), however, the error drops to under 5% for most ρ. At the highest sparsity levels, the approximated $L^{(t)}_\mathrm{ReLU}$ becomes a near-perfect predictor of empirical loss over indefinitely long sequences for all but the highest values of ρ.

4.1.3 A Geometric Interpretation of the Approximation to $L^{(t)}_\mathbf{ReLU}$

The expression we have obtained resembles (**), but the inclusion of the ReLU activation has a significant impact on its geometric interpretation. Since the model now only produces output for vectors that have a positive projection onto $\boldsymbol{W}_\mathrm{y}$, all $\boldsymbol{w}_s$ vectors in the half-space opposite $\boldsymbol{W}_\mathrm{y}$ do not contribute to projection interference.

In fact, in the extremely sparse regime (where composition interference becomes negligible), this half-space essentially becomes *interference-free*. This reveals a remarkable incentive for the model to take advantage of this phenomenon by packing as many $\boldsymbol{w}_s$ vectors into this half-space as possible. This is illustrated in figure 4.2 on the following page.

To verify this hypothesis empirically, we train linear RNNs with ReLU read-outs on the k-delay task at various levels of sparsity. Note that, due to the linear nature of the recurrence, arrangements like the one in figure 4.2(a) are not generally possible to achieve exactly in linear RNNs: no combination of rotation, reflection, scaling and shearing can be iterated to create such an arrangement of vectors for arbitrary k. Despite this, the results in figure 4.3 on page 31 confirm our hypothesis that, when sparsity is high, models learn to minimise projection interference by sending their largest activations through the interference-free half-space.



(a) Ideal arrangement for ReLU read-out

(b) Familiar solution for linear read-out

Figure 4.2: ReLU read-out creates an interference-free half-space. Both figures show a shaded half space opposite to $\boldsymbol{W}_y$ that becomes interference-free in the extremely sparse regime. In (a), we imagine the arrangement incentivised by this: ideally, only the output feature $\boldsymbol{w}_k$ lies outside the interference-free half-space, as it should align with $\boldsymbol{W}_y$ to minimise task error; all other $\boldsymbol{w}_s$ vectors should lie in the interference-free half-space so that they have no effect on the loss. In (b), we recall the arrangement learned by linear models with linear read-outs. If a ReLU read-out is introduced, this arrangement will no longer be optimal because the interference-free half-space is not being optimally used; projection interference could be significantly reduced by moving the largest vectors to the interference-free half-space.

For $k \leq 3$, we observe almost ideal solutions in the extremely sparse regime, with only the output feature represented outside the interference-free half-space. This is exactly as predicted from the geometric interpretation of $L^{(t)}_{\text{ReLU}}$. For higher values of k, the linear recurrence is more limiting: it forces the $\boldsymbol{w}_s$ vectors to be spaced "equally" along an elliptical spiral [25, p.317] and so there is no way for $k+1 \geq 5$ task-relevant features to have non-negligible norm without incurring some projection interference.[1] Thus we see an approximation of the strategy, in which the largest $\boldsymbol{w}_s$ vectors occupy the interference-free half-space while smaller vectors lie outside of it. Within the constraints of linear recurrence, this strategy still minimises projection interference by exploiting the interference-free half-space.

Unsurprisingly, the optimal spectral radius increases with k, regardless of sparsity. This is simply because for larger k, the model must hold inputs in its memory for more time steps. Concretely, if ρ is too small, then the magnitude of $\boldsymbol{w}_k$ in the hidden state will be negligible relative to the other $\boldsymbol{w}_s$ and so any task-relevant signal will be completely overpowered by projection interference from other features. Therefore, for optimal performance, ρ must increase with k. More subtle is how the optimal ρ varies as sparsity is increased. The results in figure 4.3 suggest that ρ increases up to some critical sparsity and then begins decreasing.

4.1.4 Observing a Phase Change in the Optimal Geometry

The results follow this pattern more broadly, hinting towards a phase change: for each k, the optimal arrangement in the dense regime ($p = 0$) resembles the solutions found by purely linear models with linear read-outs, as seen in figure 3.2. In these cases, despite the half-space opposite $\boldsymbol{W}_y$ being free from projection interference, it seems that the optimal strategy does not take advantage of it. We speculate that this is because composition interference is a major factor in the dense regime and that if the model were to place large activations in the half-space opposite to $\boldsymbol{W}_y$, those activations would almost always overpower any activation of the opposite-facing

[1] Intuitively, there is no way to stretch, shear or rotate an origin-centred regular pentagon (or higher polygon) such that the resulting shape has just one vertex above the x-axis.

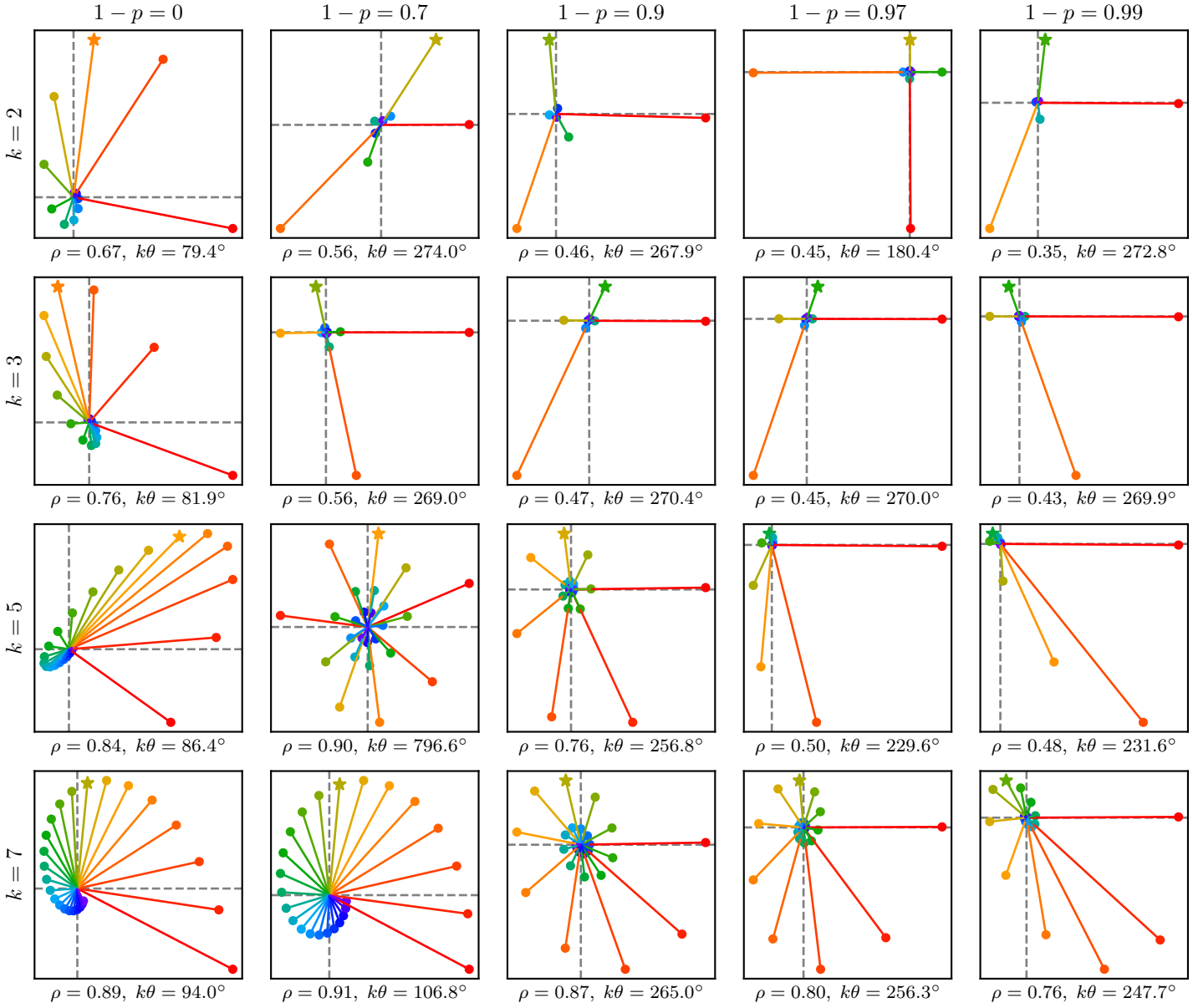


Figure 4.3: Solutions to k-delay learnt by linear models with ReLU read-out. Rows correspond to different values of k and columns correspond to different sparsity levels. For each combination of k and sparsity level, 100 linear models with 2-dimensional $\boldsymbol{W}_{\text{h}}$ and ReLU read-out were trained on 10,000 input sequences of length 25. The $\boldsymbol{w}_s$ vectors of the best-performing model (as measured by lowest EMA training loss) are plotted after applying a conformal linear transformation such that the y-component of each $\boldsymbol{w}_s$ is a_s, and $\boldsymbol{w}_0$ points towards positive x. Thus the interference-free half-space is simply given by $y < 0$. As before, the output feature, $\boldsymbol{w}_k$, is marked with a star. Note that the plots vary significantly in scale, so it is not meaningful to compare the magnitude of a particular $\boldsymbol{w}_s$ vector between different plots.

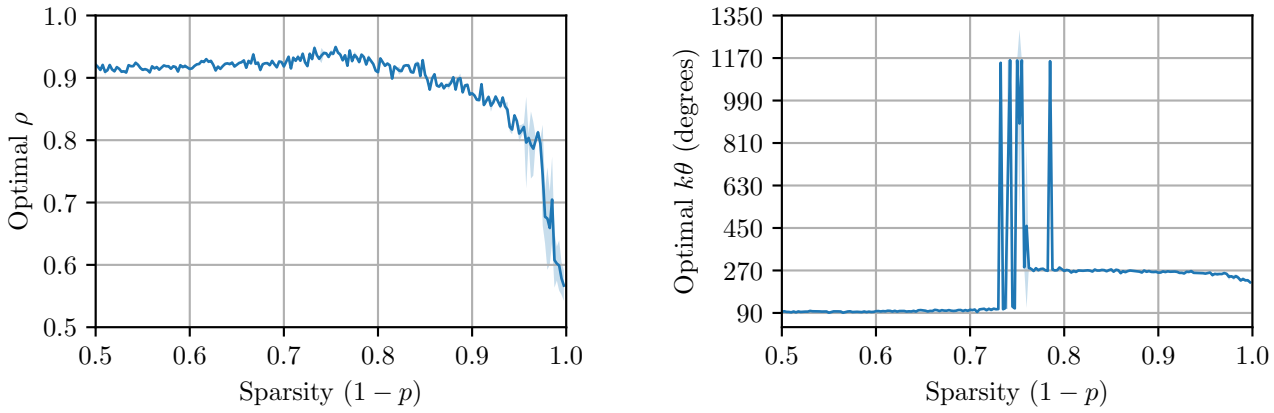


Figure 4.4: A geometric phase change in a linear model with ReLU read-out. At each sparsity level, 1000 linear models with $\boldsymbol{W}_{\text{h}} \in \mathbb{R}^{2\times 2}$ and ReLU read-out were trained over 10,000 sequences of length 25 to perform the 7-delay task. For the 10 best-performing (top 1%) models in terms of EMA training loss, the spectral radius ρ and angle $k\theta$ were calculated. The plots show the mean and standard deviation (shaded) of these values among these models. Note that the standard deviation is very low at almost all sparsity levels, implying that the top 1% of models all learnt very similar representations in almost every case; in particular, this means that the unstable regime is not an artefact of a few anomalous results but the result of a genuine metastable state.

output feature, essentially causing the model to output nothing.

On the other hand, when inputs become sufficiently sparse, the optimal strategy switches to take advantage of the interference-free half-space. Here, non-zero inputs are rare enough that most of them never interact with each other in the hidden state, meaning that there is usually no more than one input "in play". As a result, it is safe to exploit the interference-free half-space because the kind of event that made it too costly in the dense regime is extremely unlikely to occur in the sparse regime.

We now investigate the nature of the transition between these two situations. The most noticeable difference in representational geometry between the dense and sparse cases is the angular distribution of $\boldsymbol{w}_s$ vectors: in the dense regime, the task-relevant features $\boldsymbol{w}_0, \ldots, \boldsymbol{w}_k$ all exist within a cone of approximately 90°. This means that, only a quarter of the plane is being used to represent task-relevant features, while the remaining space is given to historical features. Under higher sparsity, the model then switches to an arrangement where the task-relevant features span approximately 270°, or three-quarters of the plane.

An empirical measurement of this angle is reported alongside ρ in figure 4.3 by $k\theta$, where θ is extracted from the complex conjugate eigenvalues $\lambda_{1,2} = \rho e^{\pm i\theta}$ of $\boldsymbol{W}_{\text{h}} \in \mathbb{R}^{2\times 2}$. By performing a fine-grained sweep over sparsity (figure 4.4), we uncover a clear *phase change*. Below a certain critical sparsity, the optimal value of $k\theta$ is relatively stable around 90°; then, up to a second critical sparsity level, the value is highly unstable; finally, $k\theta$ collapses again to a stable value of approximately 270°. We also observe that the optimal ρ slowly increases until around the same critical sparsity, beyond which it begins to decrease.

The existence of an unstable regime is a notable difference from the phase changes observed by [3] in feed-forward models. It suggests the existence of a transient third strategy with high $k\theta$ that is only relevant as part of the transition between the dense and sparse regimes, analogous to a metastable state. The plot for $k = 5$, $1 - p = 0.7$ in figure 4.3 appears to be an example of this state, with $k\theta = 796.6°$. More broadly, if the existence of this regime generalises beyond our toy setup, it would have a profound implication: even if we could individually understand all the geometric strategies available to an RNN, there may exist a range of sparsities within which we cannot reliably predict *which* strategy a model will actually learn.

4.2 Non-Linear Recurrence

So far, we have studied RNNs with linear recurrence. As seen in figure 4.3, such models have limited expressivity because their dynamics are constrained to a particular form, such as an elliptical spiral. In this section, we study models with a simple non-linear recurrence and find that, in the sparse regime, they consistently implement the ideal strategy of packing as many $\boldsymbol{w}_s$ vectors into the interference-free half-space as possible.

4.2.1 Introducing a ReLU Non-Linearity in the Recurrence

Applying a non-linearity to the hidden state immediately breaks the linear representation hypothesis: each feature would be represented along a (not necessarily smooth) "curve" rather than a straight line. This massively complicates the study of non-linear recurrence in general.

We therefore proceed with a recurrent setup for which, in the extremely sparse regime, the linear representation hypothesis becomes provably true. We begin with an observation that the standard non-linear recurrence with ReLU activation,

$$\boldsymbol{h}_0 = \mathbf{0}, \qquad \boldsymbol{h}_t = \mathrm{ReLU}(\boldsymbol{W}_{\mathrm{x}} x_t + \boldsymbol{W}_{\mathrm{h}} \boldsymbol{h}_{t-1}),$$

is functionally equivalent (up to a re-labelling of what we call $\boldsymbol{h}_t$) to

$$\boldsymbol{h}_0 = \mathbf{0}, \qquad \boldsymbol{h}_t = \boldsymbol{W}_{\mathrm{x}} x_t + \boldsymbol{W}_{\mathrm{h}} \,\mathrm{ReLU}(\boldsymbol{h}_{t-1}).$$

We then define the model output as $y_t = \mathrm{ReLU}\left(\boldsymbol{W}_{\mathrm{y}}^{\mathsf{T}} \boldsymbol{h}_t\right)$, reading out from the pre-activation hidden state. We acknowledge that this is a departure from the traditional RNN architecture, but we argue that it is a minor one: only the output mechanism is affected; the recurrent path in the model remains unchanged. Although this architecture is no longer literally equivalent to a standard RNN, it is still a recurrent model. Our work focuses primarily on understanding the relationship between the general concepts of recurrence, memory, sparsity and superposition, rather than on particular model architectures. As such, we believe this approach, although not typical, will still provide generalisable insights about non-linear recurrence.

4.2.2 Unrolling the Hidden State of a Non-Linear RNN Under High Sparsity

For the case of linear recurrence, we have shown that for sufficiently sparse input sequences, we can assume that there is at most one non-zero input "in play" within a model's hidden state at any given time. Fundamentally, this was based on the idea that there exists a "memory window" of length T such that inputs older than T time steps cannot contribute significantly to the current hidden state. This essentially arose from the proof in section 3.3.2 that RNNs with linear recurrence must satisfy the echo state property in order to achieve reasonable loss.

We argue that a similar "memory window" should be expected in models with non-linear recurrence. Although it is much harder to prove that the echo state property is a requirement for good performance in non-linear models, there exists plenty of evidence for the reverse statement: echo state networks are, by definition, non-linear models that satisfy the echo state property [7] and have been shown to achieve strong performance on a variety of sequential processing tasks [26]. Clearly, the echo state property is not incompatible with strong task performance and it is not implausible that the linear result – that the echo state property is actually required for tasks like k-delay – carries over to the case of non-linear recurrences, based on these empirical observations. Therefore, we intuitively expect our reasoning to hold for non-linear recurrence: if the input sequence is made sufficiently sparse, we can approximate the model's behaviour by ignoring situations where two or more inputs are non-zero.

Unrolling the hidden state, we see that an analysis of the general case is intractable:

$$\boldsymbol{h}_t = \boldsymbol{W}_{\mathrm{x}} x_t + \boldsymbol{W}_{\mathrm{h}} \,\mathrm{ReLU}(\boldsymbol{h}_{t-1})$$

$$
\begin{aligned}
&= \boldsymbol{W}_{\mathrm{x}} x_t + \boldsymbol{W}_{\mathrm{h}} \operatorname{ReLU}(\boldsymbol{W}_{\mathrm{x}} x_{t-1} + \boldsymbol{W}_{\mathrm{h}} \operatorname{ReLU}(\boldsymbol{h}_{t-2})) \\
&= \boldsymbol{W}_{\mathrm{x}} x_t + \boldsymbol{W}_{\mathrm{h}} \operatorname{ReLU}(\boldsymbol{W}_{\mathrm{x}} x_{t-1} + \boldsymbol{W}_{\mathrm{h}} \operatorname{ReLU}(\boldsymbol{W}_{\mathrm{x}} x_{t-2} + \boldsymbol{W}_{\mathrm{h}} \operatorname{ReLU}(\boldsymbol{h}_{t-3}))) \\
&\vdots \\
&= \boldsymbol{W}_{\mathrm{x}} x_t + \boldsymbol{W}_{\mathrm{h}} \operatorname{ReLU}(\boldsymbol{W}_{\mathrm{x}} x_{t-1} + \boldsymbol{W}_{\mathrm{h}} \operatorname{ReLU}(\cdots \boldsymbol{W}_{\mathrm{h}} \operatorname{ReLU}(\boldsymbol{W}_{\mathrm{x}} x_2 + \boldsymbol{W}_{\mathrm{h}} \operatorname{ReLU}(\boldsymbol{W}_{\mathrm{x}} x_1)) \cdots)).
\end{aligned}
$$

Suppose, however, that for some s, only x_{t-s} is non-zero and all other inputs are assumed to be zero, as per our approximation. Recalling that in our setup, x_{t-s} is scalar and non-negative, the hidden state simplifies as follows:

$$
\begin{aligned}
\boldsymbol{h}_t &= \boldsymbol{W}_{\mathrm{x}}(0) + \boldsymbol{W}_{\mathrm{h}} \operatorname{ReLU}(\boldsymbol{W}_{\mathrm{x}}(0) + \boldsymbol{W}_{\mathrm{h}} \operatorname{ReLU}(\cdots \boldsymbol{W}_{\mathrm{h}} \operatorname{ReLU}(\boldsymbol{W}_{\mathrm{x}} x_{t-s} + \boldsymbol{W}_{\mathrm{h}}(0)) \cdots)) \\
&= \boldsymbol{W}_{\mathrm{h}} \operatorname{ReLU}(\boldsymbol{W}_{\mathrm{h}} \operatorname{ReLU}(\cdots \boldsymbol{W}_{\mathrm{h}} \operatorname{ReLU}(\boldsymbol{W}_{\mathrm{x}} x_{t-s}) \cdots)) \\
&= \boldsymbol{W}_{\mathrm{h}} \operatorname{ReLU}(\boldsymbol{W}_{\mathrm{h}} \operatorname{ReLU}(\cdots x_{t-s} \boldsymbol{W}_{\mathrm{h}} \operatorname{ReLU}(\boldsymbol{W}_{\mathrm{x}}) \cdots)) \\
&= x_{t-s} \underbrace{\boldsymbol{W}_{\mathrm{h}} \operatorname{ReLU}(\boldsymbol{W}_{\mathrm{h}} \operatorname{ReLU}(\cdots \boldsymbol{W}_{\mathrm{h}} \operatorname{ReLU}(\boldsymbol{W}_{\mathrm{x}}) \cdots))}_{\boldsymbol{w}_s} \\
&= \boldsymbol{w}_s x_{t-s}.
\end{aligned}
$$

Though it is not analytically possible to find a simplified expression for the vector $\boldsymbol{w}_s$, it nevertheless is the direction in which the feature x_{t-s} is represented in the limit of sparsity, as $p \to 0$. Hence, in the extremely sparse regime, the linear representation hypothesis holds for this model. This is not a trivial result – it relies on both the piecewise linearity of ReLU for non-negative inputs and our definition of $\boldsymbol{h}_t$ as the hidden state prior to application of ReLU.

4.2.3 Non-Linear Models Learn to Exploit the Interference-Free Half-Space

We have shown that in the extremely sparse regime, the model's hidden state collapses to a linear representation of temporal features that is read out by projection onto $\boldsymbol{W}_{\mathrm{y}}$ with ReLU activation. In other words, this is no different to the situation discussed in section 4.1.1 and so we have already developed the necessary theory (such as the concept of the interference-free half-space) to understand the model's behaviour. The only difference in the non-linear case is that the model has increased expressivity, meaning that it is able to achieve more optimal arrangements of $\boldsymbol{w}_s$ vectors.

In figure 4.5 on the next page, we verify that the geometric interpretation of $L^{(t)}_{\mathrm{ReLU}}$ developed in section 4.1.3 carries over to the non-linear model under high sparsity. Indeed, at the highest sparsity level ($1 - p = 0.99$), for each k, the best performing model packs the k intermediate features into the interference-free half-space and only $\boldsymbol{w}_k$ lies outside of it, exactly as predicted from the approximated loss $L^{(t)}_{\mathrm{ReLU}}$.

In many cases, we actually see most of the vectors compressed into a single quadrant figure 4.6 on the following page. This occurs because the ReLU non-linearity creates a *privileged basis* in the hidden state: activations in the $x, y \geq 0$ quadrant are unaffected by ReLU, while activations in the other quadrants will immediately have one or both (in the case of $x, y < 0$) of their coordinates set to zero in the next time step. As a result, the $x, y \geq 0$ is the only quadrant in which the model can implement full-rank dynamics over multiple time steps: activations in other quadrants are either immediately mapped to the $x, y \geq 0$ quadrant or are forced to be lower rank (e.g. shrinking along a line). This behaviour is seen in many of the plots in figure 4.5, especially in the bottom left, in cases of high k and low sparsity.

This may seem restrictive, but we posit that this effect of the ReLU activation is actually what makes the non-linear model so much more expressive than the linear model. Specifically, a model with linear recurrence can only implement *smooth forgetting* with spectral radius $\rho < 1$; an input's contribution to the hidden state shrinks over time (often becoming negligible within a few time steps) but never truly disappears. On the other hand, the ReLU activation makes it

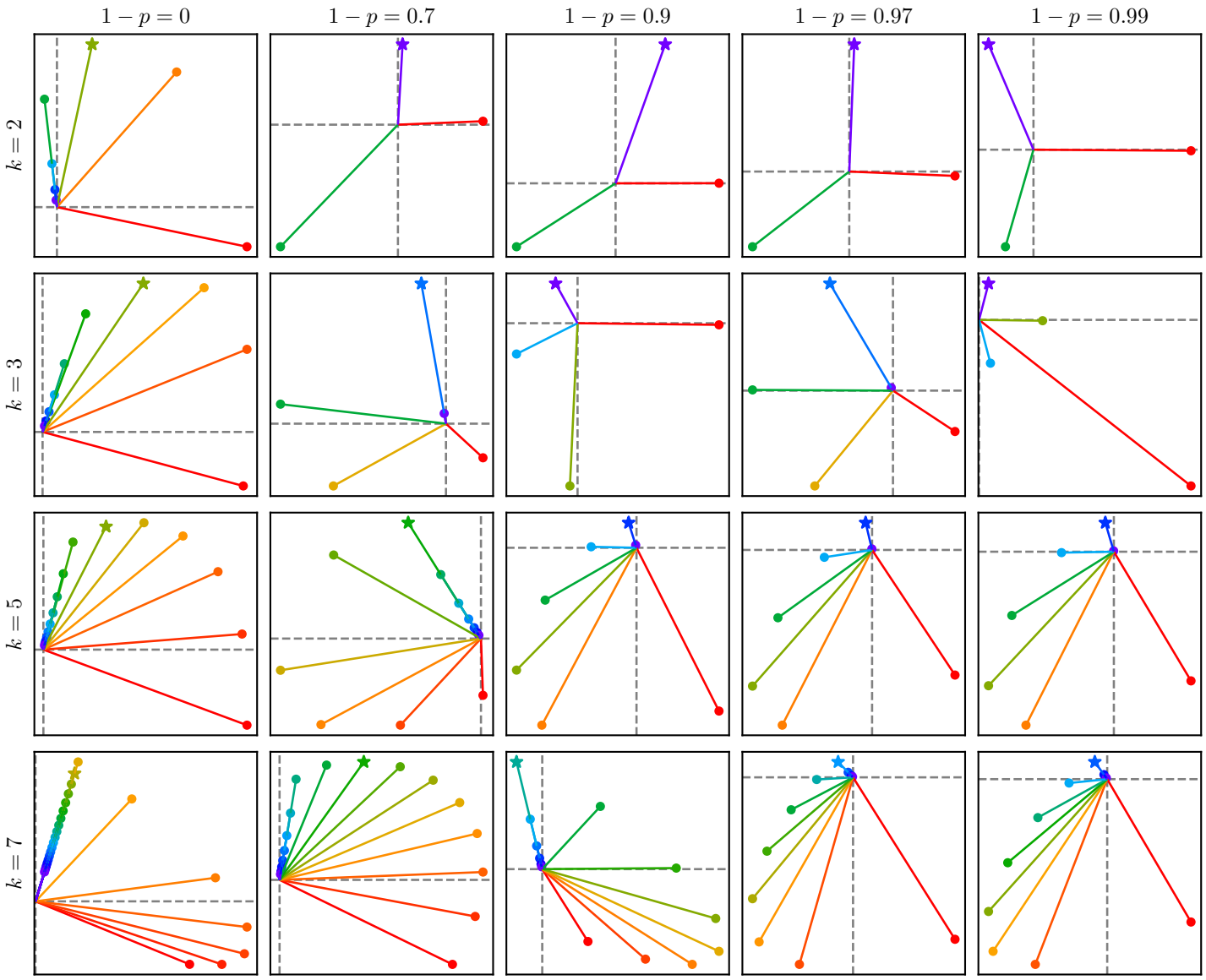


Figure 4.5: Solutions to k-delay learnt by non-linear models with ReLU read-out. Rows correspond to different values of k and columns correspond to different sparsity levels. For each combination of k and sparsity level, 100 non-linear models with 2-dimensional $\boldsymbol{W}_{\mathrm{h}}$ and ReLU read-out were trained on 10,000 input sequences of length 25. The $\boldsymbol{w}_s$ vectors of the best-performing model (as measured by lowest EMA training loss) were computed and plotted after applying a conformal linear transformation such that the y-component of each $\boldsymbol{w}_s$ is a_s, and $\boldsymbol{w}_0$ points towards positive x. Thus the interference-free half-space is simply the region defined by $y < 0$. As before, the output feature, $\boldsymbol{w}_k$, is marked with a star.

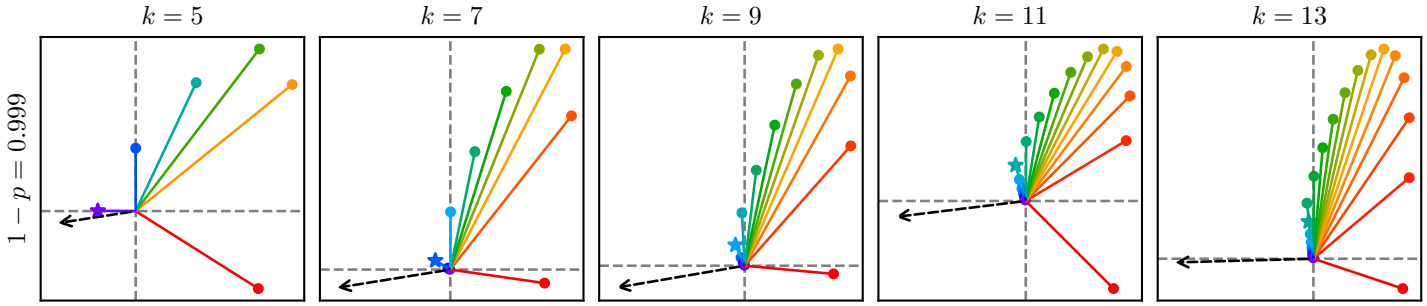


Figure 4.6: Non-linear recurrence creates a privileged basis. The experimental setup is identical to figure 4.5, but with higher sparsity and k. The axes in each plot are basis-aligned, meaning that each axis represents the activation of a neuron in the hidden state. $\boldsymbol{W}_{\mathrm{y}}$ is plotted as a dashed black arrow.

possible to immediately forget a feature by sending it to the $x, y < 0$ quadrant (or the higher-dimensional equivalent). As a result, the non-linear model can rely on the ReLU activation to implement *sharp forgetting*, which enables the model to represent only the task-relevant features. Furthermore, since it no longer relies on ρ to implement smooth forgetting, these task-relevant features can be represented with larger magnitudes in the hidden state.

Viewed another way, the ReLU activation function actually makes it far *easier* for a model to satisfy the echo state property if desired. Since inputs can be immediately forgotten once they are no longer required, the "memory window" can now be a sharp boundary. In figure 4.5, we see this implemented in the high sparsity regime for small k: models often learn to represent the $k + 1$ task-relevant features and nothing more.

Note that this behaviour is not seen for higher values of k; this is because a 2-dimensional non-linear RNN is simply not expressive enough to be able to pack many $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{k-1}$ tightly in the $x, y \geq 0$ quadrant and then transform $\boldsymbol{w}_{k-1}$ all the way to the $x, y < 0$ quadrant (where the desired behaviour would be for it to be output as $\boldsymbol{w}_k$ and then immediately forgotten).

Chapter 5

Discussion

The problem of understanding *why* models learn the internal representations that they do can be approached from two directions. One idea is to train lots of models under various conditions and analyse the empirical results. The benefit of this approach is that we can be relatively sure that the solutions we observe are near-optimal. The downside is that this approach does little to reveal *how* or *why* the optimiser arrived at those optimal solutions.

For the most part, this is the approach taken by [3] to develop the original theory of superposition in neural networks: the authors conduct a series of experiments that demonstrate what superposition *looks like* and then present (reasonable) hypotheses about why it looks that way.

The problem with applying this approach to the study of representations in RNNs is that, as we have seen, there are many unrelated forces that interact to determine what the optimal solution looks like. Even for linear recurrent models trained on a basic recall task, the optimal representation is determined by a combination of the sparsity and mean of the data, the specific memory requirement of the task (k), the effects of projection and composition interference and the underlying constraint on any learnt representation to be achievable by a linear dynamical system. It seems very difficult to look at trained RNNs and cleanly attribute each observed pattern to just one or two of these factors. For example, it seems much harder to detect the existence of the mean-correction term from just empirical observations.

In the study of feed-forward models, the authors of [3] were able to avoid issues like this by enforcing symmetries (such as $\boldsymbol{W}_{\mathrm{y}} = \boldsymbol{W}_{\mathrm{x}}$) that simplified the setup; such tricks are harder to implement for RNNs, as often the only way for an RNN to perform non-trivial computation is to break these kinds of symmetries. This makes the starting point for RNNs far more complex than that for feed-forward models.

In this work, we have demonstrated the feasibility and power of the alternative, bottom-up approach in which we build up the interpretation from a purely theoretical foundation. This way, we can *guarantee* the existence of incentives like mean-correction, even if they are difficult to detect in practice. The strength of our approach in the linear case is that there is absolutely no guesswork involved – the incentives we identified fall directly out of an exact expression for the loss. This is promising, as it establishes another proof-of-concept (alongside works like [6]) of a purely analytical approach to mechanistic interpretability.

Of course, this theoretical approach quickly became intractable once we considered any kind of non-linearity, but it was somewhat surprising to see just how far the intuitions we developed to understand the linear model carried over into more complex cases. In particular, our proposed distinction between projection interference and composition interference became the core framework for understanding the non-linear models and motivating the existence of the interference-free half-space. Again, this is highly promising, as it implies that even if further development of the theory becomes intractable, the concepts established here can be used in future work to understand empirical results. As such, our hope is that theoretical work like this can provide the necessary framework to make the empirical approach more viable for RNNs.

Limitations of Our Approach

Due to this bottom-up approach, the main limitation of our work is that it focuses entirely on a very simple toy task and opts for depth of analysis over breadth. While this has led to the development of a novel theoretical framework that shows some promise of wider generalisation, it does not capture the full range of sequential processing tasks. For example, real-world tasks will often require an RNN to perform conditional processing, where the model steps through entirely different sequences of future hidden states depending on the current input.

Related to this is the issue that we have focused solely on the case of scalar input sequences. This was crucial for developing a sound theory of temporal superposition, but we have not considered the more general *spatiotemporal* interactions. In the "future work" section below, we discuss some preliminary ideas related to this and outline a hypothesis that could be investigated in future work.

Another issue is that we have largely focused on the case of 2-dimensional hidden states. This is convenient for visualising the internal representations of models and building intuitions about how they form, but it is possible that there exist subtle differences in higher dimensions. We do suspect, however, that most results other than the "spiral sink" (which is specifically a 2-dimensional arrangement) should generalise to higher dimensions, as we generally only make the 2D assumption for the purpose of visualisation and empirical validation; the underlying theory makes no assumptions about the dimensionality of the hidden state.

Finally, our theoretical analysis relies on the assumption that temporal features are independently distributed. As discussed in section 3.2.2, this is not a realistic assumption there is likely more to be uncovered about both linear and non-linear RNN representations once this assumption is weakened or removed.

In summary, this work suffers from many of the same limitations as other "toy models" research: by definition, this research direction focuses on extracting interesting, potentially generalisable insights from very simplified and abstracted setups.

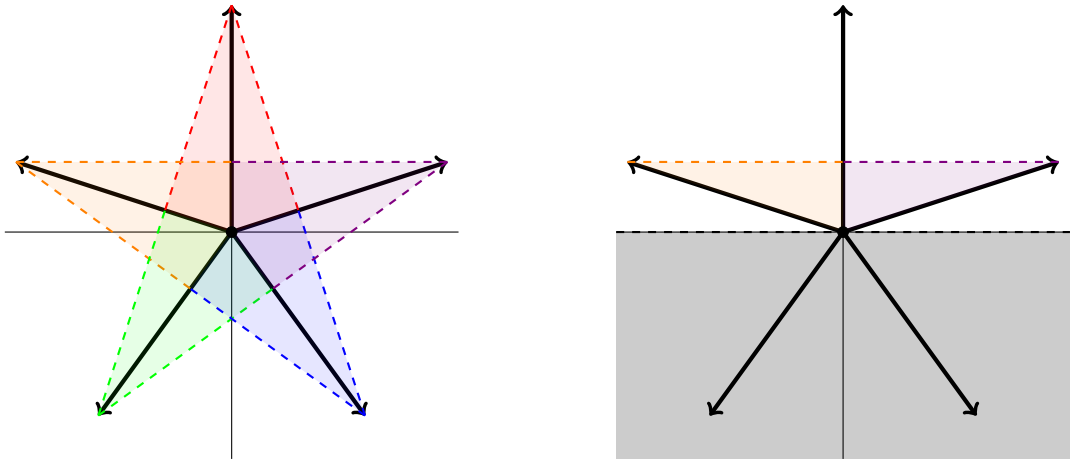
Future Work

Much of the potential for future work lies in resolving the limitations mentioned above. In particular, it may be possible to generalise the loss decomposition beyond i.i.d. input sequences – for example, studying the k-delay task under the assumption that the input sequence can be modelled by a Markov chain may still be tractable and would be substantially more convincing in terms of generalisation to real-world data.

Similarly, one might consider how our findings generalise to RNNs that are trained to simulate arbitrary finite-state machines; the k-delay task is a trivial case of this. Again, this would make for a significant generalisation of the theory to more complex sequential processing tasks. This may involve further refinement to our definitions, such as a more advanced definition of what we consider to be a "temporal feature". Mechanistic interpretability has been described as a *pre-paradigmatic* field, in that it lacks consensus on these core definitions; some of our definitions will likely need to be adapted in order to make progress.

One surprising result that we did not investigate further was the existence of an unstable regime in between the dense and sparse regimes (**??**). Future work could seek to confirm whether a metastable state truly exists in this regime and, if so, what its geometric properties are. Additionally, it would be highly interesting to see if the geometry of the model can be reliably predicted within this regime, or if there truly is a regime for which an RNN's internal representation is unpredictable.

Finally, we propose an interesting hypothesis that arises directly from our study of temporal superposition: **temporal superposition may be fundamentally "cheaper" than spatial superposition**. The reasoning for this is as follows. Generally speaking, in both feed-forward and recurrent models, we have one read-out vector per spatial feature. This means that, in a



(a) Spatial projection interference

(b) Temporal projection interference

Figure 5.1: Projection interference in spatial and temporal features. In (a), we see projection interference as it would apply in the feed-forward model studied by [3]. Each feature is perfectly aligned with its own dedicated read-out vector, so adjacent features project onto each other. Since each feature corresponds to a read-out vector, every feature in this arrangement experiences projection sparsity equally, and no region is completely free of projection interference. In (b), we see the same polygonal arrangement but in the case of temporal features, where the read-out is along the positive y-axis. As a result, only two features cause projection interference and there exists a half-space free of projection interference.

feed-forward model, the ratio between the number of read-out vectors and the number of spatial features is one-to-one.

On the other hand, in many sequential processing tasks such as k-delay, there are plenty of task-relevant temporal features that are not intended to be read out – as we have seen, these are the intermediate temporal features. Thus, in such sequential processing tasks, the task-relevant features greatly outnumber the read-out vectors and so the equivalent ratio is one-to-many.

Recalling that projection interference, by definition, can only occur in the presence of a read-out vector, we hypothesise that temporal features are fundamentally cheaper to represent than spatial features as a result of this difference in ratio. We have already seen an instance of this in our discussion of the interference-free half-space, as depicted in figure 5.1. While it is possible that this phenomenon is merely a quirk of the particular setup we have studied, it seems rather plausible that there *is* a genuine difference between the purpose of a spatial feature and that of a temporal feature, which would lead to different representational costs. If such a difference exists, it would have significant implications for the internal representations of RNNs that operate on more than one spatial feature (i.e. vector-valued sequences).

Chapter 6

Conclusion

In this work, we have built geometric intuitions about the internal representations of linear and non-linear RNNs from a purely theoretical basis. We have introduced a distinction between two separate kinds of interference: projection interference and composition interference, and shown that these phenomena behave very differently to in linear RNNs optimised for the k-delay task. We found an exact decomposition of the expected loss incurred by any linear model on this task and gave a geometric interpretaion of what each term in this decomposition incentivises.

In the study of linear models with non-linear read-out, we derived an approximation to the loss function for this task under the assumption of extremely sparse input and demonstrated that the concepts of projection and composition interference could be used to predict and explain the internal representations of these models for the k-delay task. In particular, we motivated the existence of a half-space in activation space that becomes completely free of all interference in the extremely sparse regime and verified that this encourages models to pack their largest activations into this region.

Finally, we introduced a modified non-linear RNN that provably satisfies the linear representation hypothesis in the limit of perfect sparsity ($p \to 0$). Our theoretical framework almost exactly predicted the internal representations learnt by these models.

Our work represents initial progress towards a general theoretical framework for understanding the internal representations of RNNs and a proof-of-concept for an analytical approach to RNN interpretability. We believe there is significant potential for further development and validation of this framework in more general settings.

Bibliography

[1] C. Olah, N. Cammarata, L. Schubert, G. Goh, M. Petrov, and S. Carter, "Zoom in: An introduction to circuits," *Distill*, 2020. [Online]. Available: https://distill.pub/2020/circuits/zoom-in

[2] T. Bricken, A. Templeton, J. Batson, B. Chen, A. Jermyn, T. Conerly, N. Turner, C. Anil, C. Denison, A. Askell, R. Lasenby, Y. Wu, S. Kravec, N. Schiefer, T. Maxwell, N. Joseph, Z. Hatfield-Dodds, A. Tamkin, K. Nguyen, B. McLean, J. E. Burke, T. Hume, S. Carter, T. Henighan, and C. Olah, "Towards monosemanticity: Decomposing language models with dictionary learning," *Transformer Circuits Thread*, 2023, https://transformer-circuits.pub/2023/monosemantic-features/index.html.

[3] N. Elhage, T. Hume, C. Olsson, N. Schiefer, T. Henighan, S. Kravec, Z. Hatfield-Dodds, R. Lasenby, D. Drain, C. Chen, R. Grosse, S. McCandlish, J. Kaplan, D. Amodei, M. Wattenberg, and C. Olah, "Toy models of superposition," *Transformer Circuits Thread*, 14 September 2022. [Online]. Available: https://transformer-circuits.pub/2022/toy_model/index.html

[4] C. Olah, "Distributed Representations: Composition & Superposition," *Transformer Circuits Thread*, 4 May 2023. [Online]. Available: https://transformer-circuits.pub/2023/superposition-composition/index.html

[5] A. Scherlis, K. Sachan, A. S. Jermyn, J. Benton, and B. Shlegeris, "Polysemanticity and capacity in neural networks," *CoRR*, vol. abs/2210.01892, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2210.01892

[6] K. Hänni, J. Mendel, D. Vaintrob, and L. Chan, "Mathematical models of computation in superposition," 2024. [Online]. Available: https://arxiv.org/abs/2408.05451

[7] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks," GMD Report 148, 2001. [Online]. Available: https://www.ai.rug.nl/minds/uploads/EchoStatesTechRep.pdf

[8] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*, 3rd ed., 2025, online manuscript released January 12, 2025. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/

[9] A. Karpathy, "The unreasonable effectiveness of recurrent neural networks," 2015. [Online]. Available: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

[10] O. Barak, "Recurrent neural networks as versatile tools of neuroscience research," *Current opinion in neurobiology*, vol. 46, pp. 1–6, 2017, iD: 272014. [Online]. Available: https://doi.org/10.1016/j.conb.2017.06.003

[11] H. Jaeger, "Short term memory in echo state networks," GMD Report 152, 2002. [Online]. Available: https://www.ai.rug.nl/minds/uploads/STMEchoStatesTechRep.pdf

[12] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016, iD: Silver2016. [Online]. Available: https://doi.org/10.1038/nature16961

[13] C. Metz, "In two moves, alphago and lee sedol redefined the future," 2016. [Online]. Available: https://www.wired.com/2016/03/two-moves-alphago-lee-sedol-redefined-future/

[14] C. Olah, A. Mordvintsev, and L. Schubert, "Feature visualization," *Distill*, 2017, https://distill.pub/2017/feature-visualization.

[15] T. Bricken, A. Templeton, J. Batson, B. Chen, A. Jermyn, T. Conerly, N. Turner, C. Anil, C. Denison, A. Askell, R. Lasenby, Y. Wu, S. Kravec, N. Schiefer, T. Maxwell, N. Joseph, Z. Hatfield-Dodds, A. Tamkin, K. Nguyen, B. McLean, J. E. Burke, T. Hume, S. Carter, T. Henighan, and C. Olah, "Towards monosemanticity: Decomposing language models with dictionary learning," *Transformer Circuits Thread*, 2023, https://transformer-circuits.pub/2023/monosemantic-features/index.html.

[16] R. Csordás, C. Potts, C. D. Manning, and A. Geiger, "Recurrent neural networks learn to store and generate sequences using non-linear representations," in *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, Y. Belinkov, N. Kim, J. Jumelet, H. Mohebbi, A. Mueller, and H. Chen, Eds. Miami, Florida, US: Association for Computational Linguistics, nov 2024, pp. 248–262. [Online]. Available: https://aclanthology.org/2024.blackboxnlp-1.17/

[17] N. Elhage, R. Lasenby, and C. Olah, "Privileged bases in the transformer residual stream," 2023. [Online]. Available: https://transformer-circuits.pub/2023/privileged-basis/index.html

[18] S. Dasgupta and A. Gupta, "An elementary proof of a theorem of Johnson and Lindenstrauss," *Random Structures & Algorithms*, vol. 22, no. 1, pp. 60–65, 2003, 20. [Online]. Available: https://doi.org/10.1002/rsa.10073

[19] G. Ballarin, L. Grigoryeva, and J.-P. Ortega, "Memory of recurrent networks: do we compute it right?" *J. Mach. Learn. Res.*, vol. 25, no. 1, Jan. 2024.

[20] R. H. R. Hahnloser, A. A. Kozhevnikov, and M. S. Fee, "An ultra-sparse code underlies the generation of neural sequences in a songbird," *Nature*, vol. 419, no. 6902, pp. 65–70, 2002, iD: Hahnloser2002. [Online]. Available: https://doi.org/10.1038/nature00974

[21] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge: Cambridge University Press, 1985.

[22] G. H. Hardy, *Divergent Series. Corrected reprint.* Oxford: Clarendon Press, 1973.

[23] O. Galor, *Discrete Dynamical Systems*. Berlin, Germany: Springer, Dec. 2007.

[24] D. M. Bradley and R. C. Gupta, "On the distribution of the sum of n non-identically distributed uniform random variables," *Annals of the Institute of Statistical Mathematics*, vol. 54, no. 3, pp. 689–700, 2002, iD: Bradley2002. [Online]. Available: https://doi.org/10.1023/A:1022483715767

[25] D. Margalit and J. Rabinoff, *Interactive Linear Algebra*. Georgia Institute of Technology: Self-published, 2019. [Online]. Available: https://textbooks.math.gatech.edu/ila/index.html

[26] P. V. Aceituno, G. Yan, and Y.-Y. Liu, "Tailoring echo state networks for optimal learning," *iScience*, vol. 23, no. 9, p. 101440, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2589004220306325

Declarations

Use of Generative AI I confirm that no part of this report, including its structure and content, was generated by AI.

Ethical Considerations There are no ethical issues associated with this project, as it is highly theoretical and abstract.

Sustainability This project is concerned with training very small models, which requires minimal energy consumption, so this work has minimal environmental impact.

Availability of Data and Materials The code repository associated with this project and its experiments is available at https://github.com/kashparty/rnns.

Appendix

A.1 Deriving an Alternative Form of (*)

$$
\begin{aligned}
L^{(t)} &= p\nu(a_k-1)^2 + p\nu\sum_{s\neq k}^{t-1} a_s^2 - 2p^2\mu^2\sum_{s\neq k}^{t-1} a_s + p^2\mu^2\left[\left(\sum_{s=0}^{t-1} a_s\right)^2 - \sum_{s=0}^{t-1} a_s^2\right] \\
&= \left(p\nu - p^2\mu^2\right)(a_k-1)^2 + \left(p\nu - p^2\mu^2\right)\sum_{s\neq k}^{t-1} a_s^2 \\
&\quad + p^2\mu^2(a_k-1)^2 + p^2\mu^2\sum_{s\neq k}^{t-1} a_s^2 - 2p^2\mu^2\sum_{s\neq k}^{t-1} a_s + p^2\mu^2\left(\sum_{s=0}^{t-1} a_s\right)^2 - p^2\mu^2\sum_{s=0}^{t-1} a_s^2 \\
&= \left(p\nu - p^2\mu^2\right)\left((a_k-1)^2 + \sum_{s\neq k}^{t-1} a_s^2\right) + p^2\mu^2\left(1 + \sum_{s=0}^{t-1} a_s^2 - 2\sum_{s=0}^{t-1} a_s + \left(\sum_{s=0}^{t-1} a_s\right)^2 - \sum_{s=0}^{t-1} a_s^2\right) \\
&= \left(p\nu - p^2\mu^2\right)\left((a_k-1)^2 + \sum_{s\neq k}^{t-1} a_s^2\right) + p^2\mu^2\left(\sum_{s=0}^{t-1} a_s - 1\right)^2
\end{aligned}
$$

A.2 Proof of $\Pr[N_\varepsilon \geq 2] \approx \mathcal{O}\left(p^2T_\varepsilon^2\right)$ for $N_\varepsilon \sim \text{Binomial}(T_\varepsilon, p)$ with small p

Suppose $N_\varepsilon \sim \text{Binomial}(T_\varepsilon, p)$. Then

$$
\begin{aligned}
\Pr[N_\varepsilon \geq 2] &= 1 - \Pr[N_\varepsilon = 0] - \Pr[N_\varepsilon = 1] \\
&= 1 - (1-p)^{T_\varepsilon} - pT_\varepsilon\,(1-p)^{T_\varepsilon - 1}.
\end{aligned}
$$

For small p, we can apply a binomial approximation to $(1-p)^n$. This gives

$$
\begin{aligned}
(1-p)^{T_\varepsilon} &\approx 1 - pT_\varepsilon + \frac{1}{2}p^2T_\varepsilon(T_\varepsilon - 1), \\
pT_\varepsilon\,(1-p)^{T_\varepsilon-1} &\approx pT_\varepsilon\left[1 - p(T_\varepsilon - 1) + \frac{1}{2}p^2(T_\varepsilon-1)(T_\varepsilon-2)\right] \\
&= pT_\varepsilon - p^2T_\varepsilon(T_\varepsilon-1) + \frac{1}{2}p^3T_\varepsilon(T_\varepsilon-1)(T_\varepsilon-2).
\end{aligned}
$$

Therefore

$$
\begin{aligned}
\Pr[N_\varepsilon \geq 2] &= 1 - (1-p)^{T_\varepsilon} - pT_\varepsilon\,(1-p)^{T_\varepsilon-1} \\
&\approx 1 - \left[1 - \frac{1}{2}p^2T_\varepsilon(T_\varepsilon-1) + \frac{1}{2}p^3T_\varepsilon(T_\varepsilon-1)(T_\varepsilon-2)\right] \\
&= \frac{1}{2}p^2T_\varepsilon(T_\varepsilon-1) + \mathcal{O}\left(p^3\right).
\end{aligned}
$$