FIXED POINTS, LEARNING, AND PLASTICITY IN RECURRENT NEURONAL NETWORK MODELS

A Dissertation

Submitted to the Graduate School
of the University of Notre Dame
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

by

Vicky R. Zhu

Robert J. Rosenbaum, Director

Graduate Program in Applied and Computational Mathematics and Statistics
Notre Dame, Indiana
April 2023

FIXED POINTS, LEARNING, AND PLASTICITY IN RECURRENT NEURONAL NETWORK MODELS

Abstract

by

Vicky R. Zhu

Recurrent neural network models (RNNs) are widely used in machine learning and in computational neuroscience. While recurrent in artificial neural networks (ANNs) share some basic building blocks with cortical neuronal networks in the brain, they differ in some fundamental ways. For example, neurons communicate and learn differently. In ANNs, neurons communicate through activations. In comparison, biological neurons communicate via synapses and signal processing along with neuron spiking behaviors. To link neuroscience and machine learning, I study models of recurrent neuronal networks to establish direct, one-to-one analogs between artificial and biological neuronal networks.

I first showed their connection by formalizing the features of cortical networks into theorems that link to machine learning activations. This work extended the traditional excitatory-inhibitory balance network theory into a "semi-balanced" state in which networks implement high-dimensional and nonlinear stimulus representations. To understand brain operations and neuron plasticity, I combined numerical simulations of biological networks and mean-field rate models to evaluate the extent to which homeostatic inhibitory plasticity learns to compute prediction errors in randomly connected, unstructured neuronal networks. I found that homeostatic synaptic plasticity alone is not sufficient to learn and perform non-trivial predictive coding tasks in unstructured neuronal network models. To further invest in learning, I derived two new biologically-inspired RNN learning rules for the fixed points of recurrent dynamics. Under a natural re-parameterization of the network model, they can be interpreted as steepest descent and gradient descent on the weight matrix with respect to a non-Euclidean metric and gradient, respectively. Moreover, compared with the standard gradient-based learning methods, one of our alternative learning rules is robust and computationally more efficient. These learning rules produce results that have implications for training RNNs to be used in computational neuroscience studies and machine learning applications.

DEDICATION

This thesis is dedicated to my parents and family friends who have been constantly encouraging and advising me during every challenging stage of my life. I am truly thankful to my devoted parents, Kegang Zhu and Changhua Liu, and my brother Philip Zhu for their unwavering support and unconditional love that made my American dream possible. A special feeling of gratitude to our family friend, William Hranchak, who always shares his wisdom, inspires me to work hard and never give up. To everyone who told me to believe in higher education, dare to dream, try hard and work harder. Everything is possible to achieve!

CONTENTS

Figures vi

Tables xv

Acknowledgments xvi

Symbols xviii

Chapter 1: Introduction 1
1.1 A Brief History of A Brain-inspired Machine Learning Architecture . 1
1.2 Recurrent Neural Networks (RNNs) 3
1.2.1 Artificial Recurrent Neural Networks (ARNNs) 3
1.2.2 Biological Recurrent Neural Networks (BRNNs) 5
1.3 Overview and Contributions 7

Chapter 2: Biological Basics and Mathematical Modeling 9
2.1 Biological Neurons 9
2.2 The Modeling Development of A Single Neuron 13
2.2.1 Leaky Integrator Model 13
2.2.2 Exponential Integrate-and-Fire (EIF) Model 15
2.2.3 Synapses-Driven Model 17
2.2.4 Mean-Field Theory 20
2.2.5 f-I Curve 21
2.3 Modeling of A Network of Neurons 23
2.3.1 Recurrent Network 24
2.3.2 Connectivity 27
2.3.3 Rate Model Approximations 28
2.3.4 Balanced Network Theory 29
2.4 Modeling Through Synaptic Plasticity and Learning 34
2.4.1 Synaptic Plasticity 35
2.4.2 Learning 39

Chapter 3: Universal Properties of Strongly Connected Networks 41
3.1 Introduction 42
3.2 Spiking Network Model Descriptions 43
3.2.1 Simulations of An Adaptive EIF Model 44
3.2.2 Simulations with Inhibitory Plasticity 46
3.3 Linear Representations in Balanced Networks 47
3.4 Nonlinear Representations in Semi-balanced Networks 50
3.4.1 Conditions Break the Classical Balanced State 50
3.4.2 Semi-balanced State in BRNNs 52
3.4.3 A Direct Correspondence to ANNs 56
3.4.4 Homeostatic Plasticity Produces "Detailed Semi-balanced" . . 59
3.5 Discussion 64

Chapter 4: Can Homeostatic Plasticity Learn to Compute Prediction Errors? 68
4.1 Introduction 69
4.2 Model Descriptions 72
4.2.1 An EIF Network Model with Homeostatic Plasticity 72
4.2.2 Simulation Parameters 73
4.2.2.1 In the Spiking Network Model 74
4.2.2.2 In the Mean-Field Rate Network Model 74
4.3 Detectable Prediction Errors After Training 75
4.3.1 Time-constant Inputs 75
4.3.2 An Analysis of Multiple Sub-populations for the Error Detection 77
4.3.2.1 Mean-field Rate Model Approximation 78
4.3.2.2 Separation of Timescales Approximation 84
4.3.3 Distributed and Time-constant Inputs 88
4.4 Undetectable Prediction Errors After Training 90
4.4.1 Time-varying Inputs 91
4.4.2 A Mean-field Explanation for the Absence of Detection 93
4.4.2.1 Timescale Assumptions of the Stimuli 93
4.4.2.2 Separation of Timescales over Mean Approximation . 94
4.4.3 Distributed and Time-varying Inputs 99
4.5 Generalization 101
4.5.1 Networks with External Input to Inhibitory Populations . . . 102
4.5.2 Networks with Increasing Mismatch Stimuli 107
4.5.3 Networks with More Sub-populations 107
4.6 Discussion 109

Chapter 5: Learning Fixed Points in Recurrent Neural Network Models 114
5.1 Introduction 115
5.2 Fixed Point Firing Rate Model and Machine Learning Tasks 117
5.2.1 Firing Rate Model Descriptions 117
5.2.2 Fixed Point Stability 118

5.2.3 Supervised Learning 118
5.2.4 Gradient Descent on the Recurrent Weight. 120
5.3 Newly Developed Learning Rules 121
5.3.1 Nonlinear Reparameterizing of the RNNs 121
5.3.2 Linear Approximation of the Reparameterized Rule 126
5.3.3 Regularization for Fixed Point Problems 130
5.4 Experiments and Results 131
5.4.1 Example 1: Linear Least Squares Problem in One-dimension. . 132
5.4.1.1 Learning Through the Gradient Descent Approach . 134
5.4.1.2 Learning Under the Reparameterized Update. 135
5.4.1.3 Learning Via Linearizing the Reparameterized Rule . 138
5.4.2 Example 2: Linear Least Squares Problem in Higher-dimensions.138
5.4.2.1 Learning Through the Gradient Descent Approach . 143
5.4.2.2 Learning Under the Reparameterized Update. 144
5.4.2.3 Learning Via Linearizing Reparameterized rule . . . 146
5.4.2.4 An Under-parameterized Linear System. 148
5.4.3 Example 3: Training Fixed Points on A Categorical Task. . . 151
5.4.3.1 Learning with Linear Activation Function 153
5.4.3.2 Learning with ReLu Activation using Support Sets . 157
5.4.3.3 Learning with a Sigmoidal Activation Function . . . 161
5.5 Discussions 163

Chapter 6: Summary and Future Directions 167
6.1 Summary 167
6.2 Future Work 169

Appendix A: Learning Fixed Points in RNNs (Corresponding to Chapter 5) . 172
A.1 Derivation of the Direct Gradient Descent Rule 172
A.2 Analysis of A Natural Reparameterization Learning Rule 177
A.3 Linearization of the New Reparameterization Rule 179
A.4 Convexity of the Cost Function for the Linear One-dimensional Model. 180
A.5 Stability Region Boundary 182
A.6 Optimal Parameters in Linear Networks 183
A.7 Learning Rules for RNNs with Nonlinear Activation Functions. 185

Bibliography 190

FIGURES

1.1 **Brain regions.** a: Brain division. b: Cortical regions with areas that are associated with several specific cognition tasks. Image is taken from Google Search. 2

1.2 **Neural network architecture in machine learning** Left: multi-layered feedforward networks (a.k.s DNN) are used to learn static representations $y = F(x)$. Middle: RNNs are used to learn functions between time series, $y(t) = \mathcal{F}[x](t)$. Right: Multi-layered RNNs that is cortical circuits alike and learn static functions as well as functions between time series. 3

1.3 **A single-unit RNN diagram**. Left: a compressed unit. Right: the unfolded version of the compressed unit. From bottom to top: time series input X_t, hidden layer h_t, and output O_t. U, V, and W are the connectivity weights of the network. Image is taken from Wikipedia RNN Search. 4

1.4 **A 4-layered biological RNN.** A biological RNN architecture to model the visual cortex from Liao and Poggio's experiments. The network contains feed-forward connection (left-to-right), feed-back connections (right-to-left) and lateral connections (looping back to same area; synonymous with recurrent connections in ARNNs terminology). [54] 6

2.1 **Neuron shapes and its structure.** a: Illustration of neuron morphologies vary in size, shape, and location. b: The basic structure of a neuron. Image is taken from Healthline Media: https://www.healthline.com/ 10

2.2 **Neuron communication diagram**. First Neuron (a.k.a. transmitting *or* presynaptic neuron). Second Neuron (a.k.a receiving *or* post-synaptic neuron). The image is taken from a Google Image search. . . 12

2.3 **EIF model simulation**. Top: a constant external input injection. Middle: the membrane potential dynamics. Bottom: a spike train simulation is to describe the middle plot. 16

2.4 **Leaky integrator Model driven by two pre-synaptic neurons**. A: excitatory spike train. B: excitatory synaptic current. C: same as A and B, except for the inhibitory neuron. E: synapse diagram with two Pre-synaptic neurons and one post-synaptic neuron. F: the membrane potential dynamic of the post-synaptic neuron. 19

2.5 **EIF model driven by a population of pre-synaptic neurons**. A: synapse diagram with 100 Pre-synaptic excitatory neurons and 25 inhibitory post-synaptic neurons. B: raster plots of pre-synaptic excitatory neurons. C: mean value of excitatory pre-synaptic neuron current. The dash line is the stationary mean. D, E: same as B and C, except for the inhibitory pre-synaptic neurons. F: the membrane potential dynamic of the post-synaptic neuron. G: the total synaptic input current by summing C and E. 22

2.6 **f-I curve for an EIF driven by Poisson synaptic input.**. A: Membrane potential of an EIF in a non-spiking regime. B: Membrane potential of an EIF in the noise-driven regime. C: An f-I curve for an EIF. The firing rate was plotted as a function of stationary mean synaptic input. Varying $\bar{I}_{syn}$ through r_e while keeping N_a and J_a unchanged. The blue and red dots correspond to the firing rate from A and B. Dotted green line is the threshold linear fit of data. 24

2.7 **EIF model in a recurrent network.** A: the network scheme with $N_x = 1000$ external neuron spikes using a Poisson process, and the recurrent network contains $N_e = 4000$ excitatory neurons and $N_i = 1000$ inhibitory ones. B, C, and E: roaster plots of 100 neurons from each group. D and F: trial-average firing rates of each post-synaptic neuron group and their mean-field rates in dashed lines. 26

2.8 **Stability of the fixed point in RNNs.** A: BRNN in Section 2.3.1 under EIF model simulations and using a mean-field approximation of rate model in Eq. 2.19 with $\tau_e = 40ms$ and $\tau_i = 20ms$ to obtain population firing rates. B: the eigenvalue of the Jacobian matrix plot, $\text{Re}\{\lambda(Jac)\} < 0$, so A has stable fixed points for each population. C: Same as in B, except for the weight matrix, W. A, B, and C are the population level. D: an RNN network with a size of $N = 100$ neurons directly under the rate model simulations. E, and F: same as in A, B, and C, except RNN contains some unstable fixed points for individual neurons in D, they are points on the right side of the zero bar in E and outside of the unit circle in F. D, E, and F are the neuron level, where W is generated from a random matrix. 30

2.9 **E-I Balanced in a recurrent network simulated from an EIF spiking model.** Top: mean current input into excitatory (in red, positive) and inhibitory (in blue, negative) population within the recurrent network. The mean total input (in black) is the summation of all the inputs, and it is bouncing around 0. Bottom: trial-average firing rates (in solid colored lines) for each neuron population and their stationary mean-field rates (in dashed colored lines). 33

2.10 **ISP from an inhibitory neuron to an excitatory neuron** A: diagram of a synaptic neuron pair from inhibition to excitation. B: the spike time delay, Δt, between two spikes. C: the change of synaptic strength, ΔJ_{ei}, as a function of Δt. 37

2.11 **ISP under an EIF spiking model in a recurrent network** A: diagram of a recurrent spiking network using the same parameter setting as in Figure 2.7. B: firing rates of dynamics from the EIF spiking model evolves through ISP: the inhibitory firing rate (in blue) pushes the excitatory firing rates (in red) towards its target rate (in black, r_0. C: a mean-field approximation rate model of B. 39

3.1 **Balanced and Semi-balanced States** a: Network Diagram. A recurrent spiking network of $N = 3 \times 10^4$ model neurons is composed of two excitatory populations ($e1$ and $e2$) and one inhibitory population (i) that receive input from two external spike train populations ($x1$ and $x2$). b: The two-dimensional space of external population firing rates represents a stimulus space. The filled triangle and circle show the two stimulus values used in d the first half of 500ms and the second half of 500ms correspondingly. c: Raster plots of 200 randomly selected spike trains from each population for two stimuli, and below is the membrane potential of one neuron from population $e1$. d: Mean input current to population $e1$, $e2$, and i from all excitatory sources ($e1, e2, x1$, and $x2$; red), from the inhibitory population (i; blue), and from all sources (black) showing approximate excitatory-inhibitory balance across stimuli in the first half of 500ms. With excess inhibition in the second half of 500ms, the semi-balance state shows that $e2$ and i population form a balanced sub-network. 45

3.2 **Firing Rates Representation in Balanced and Semi-balanced Networks** a: Network Diagram, same as in Figure 3.1 except included a linear readout R output. b: top is the neural manifold traced out by firing rates in each population in the recurrent network as external firing rates are varied across a square in stimulus space ($0 \leq r_x1, r_x2 \leq 30$), and the bottom is the readout as a function of r_x1 and r_x2 from the same simulation. c: Same as c, except this is for the semi-balanced state. All firing rates are in Hz. 49

3.3 **Detailed imbalance, balance, semi-balance, and distributed neural representations** a: Network diagram. Same as in Figure 3.1a except there is just one excitatory and one external population and an additional input $\vec{Z} = \sigma_1 \vec{Z}_1 + \sigma_2 \vec{Z}_2$. b: Histograms of input currents to all excitatory neurons averaged over the first 40s (gray, imbalanced), the next 40s (yellow, balanced), and the last 40s (purple, semi-balanced). c: Excitatory (red), inhibitory (blue), and total (black) input currents to 100 randomly selected excitatory neurons averaged over 2s time bins. During the first 40s, synaptic weights and $\sigma_1 = \sigma_2$ were fixed. During the next 40s, homeostatic iSTDP was turned on and $\sigma_1 = \sigma_2$ were fixed. During the last 40s, iSTDP was on and σ_1 and σ_2 were selected randomly every 2s. d: Firing rates of the same 100 neurons averaged over 2s bins. 61

4.1 **Prediction errors after training on time-constant inputs to multiple sub-populations.** A and B: Network diagram with "training" and "mismatch" stimuli respectively. A randomly connected, recurrent spiking neural network of $N = 5000$ neurons consisted of two excitatory sub-populations (e_1 and e_2) and one inhibitory (i) population. During the first 100s of the simulation, the network received a "training" stimulus in which e_1 and e_2 received extra external input modeling bottom-up and top-down stimuli respectively (A). Then a "mismatch" stimulus was introduced for 1s by removing the top-down stimulus to population e_2. C: Homeostatic inhibitory synaptic plasticity caused population-averaged firing rates to converge to their targets during training, but they deviated from their targets in response to the mismatch stimulus. D: The deviation of the mean firing rates from their targets (MSE_{mean}) and the mean deviation of individual neurons' firing rates (MSE_{pop}) quantify the deviation of firing rates from their targets. E and F: Raster plots (top) and membrane potential (bottom) of a random subset of neurons from population e_1. 76

4.2 **mean-field firing rate model captures the dynamics of the spiking network model.** A: Firing rates of the mean-field firing rate model defined by Eqs. (4.6) and (4.6). Compare to Figure 4.1C. B: MSE deviation of the firing rates from their targets (MSE_{mf}; light green) and the MSE with a Poisson correction ($MSE_{Poisson}$; dark green). Compare to Figure 4.1D. 80

4.3 **Slow dynamics are captured by a separation-of-timescales approximation.** A: Firing rates of the model defined by Eqs. (4.12). Compare to Figures 4.1C and 4.2A. B: MSE deviation of the firing rates from their targets (MSE_{mf}; light green) and the MSE with a Poisson correction ($MSE_{Poisson}$; dark green) from the model defined by Eqs. (4.12). Compare to Figures 4.1D and 4.2B. C: Deviation of the inhibitory weights, w_{ai}, from the fixed point values given in Eqs. (4.14). 84

4.4 **Prediction errors after training on distributed time-constant inputs.** Same as Figure 4.1 except bottom-up and top-down inputs were modeled as distributed stimuli using multivariate Gaussian inputs vectors (Eq. (4.18)). 89

4.5 **Undetectable prediction errors in a model with time-varying stimuli.** A and B: Network schematic. Same as Figure 4.1A except for the magnitude of the top-down and bottom-up stimuli were multiplied by the same time-varying signal, $c(t)$. C-F: Same as Figure 4.1C-F except we additionally plotted the mean excitatory firing rates (black curve in C). 91

4.6 **Mean-field rate model with time-varying stimuli.** Same as Figure 4.2 except using the time-varying stimuli from Figure 4.5. 94

4.7 **Slow dynamics captured by a separation of timescales in a model with time-dependent stimuli.** Same as Figure 4.3 except using the time-varying stimuli from Figure 4.5A-B. 94

4.8 **Schematic illustrating why mismatch responses are detectable after training on time-constant, but not time-dependent stimuli.** A: Schematic representing inputs to the network in a model with time-constant stimuli. Training stimuli occupy a single point in (U, V) space (purple dot). The deviation of firing rates from their targets on any particular trial is approximately proportional to the distance of the input from its value during training (Eq. (4.15)). Since the mismatch stimulus (orange dot) is far from the matched, training stimulus, firing rates deviate from their target in response to the mismatched stimulus (as seen in Figures 4.1–4.3). B: Schematic representing inputs to the network in a model with time-varying stimuli. Training stimuli (purple dots) vary in (U, V) space along a predictable line. The mismatched stimulus lies far from this line. However, the deviation of firing rates from their targets on any particular trial is approximately proportional to the distance of the input from its *mean* value during training (Eq. (4.15)). Since the distance between the mismatch input (orange dot) and the mean training stimulus (purple x) is similar to the typical distance between the individual training stimuli (purple dots) and the mean training stimulus (purple x), the deviation of the firing rates from their targets is similar for matched and mismatched stimuli. 97

4.9 **Prediction errors after training on distributed time-dependent inputs.** Same as Figure 4.4 except bottom-up and top-down inputs were time-dependent, as described by Eqs. (4.29)–(4.30). 100

4.10 **Similar results are obtained with input to inhibitory populations.** A, B: Network schematics. Same as Figure 1A, B except top-down external input was provided to the inhibitory population as well. C, D: Same as Figure 1C, D except top-down external input was provided to the inhibitory population as well. E, F: Same as Figure 4.5C, D except top-down external input was provided to the inhibitory population as well. 103

4.11 **Similar results are obtained with distributed inputs to inhibitory populations.** A, B: Network schematics. Same as Figure 4A, B except distributed external input was provided to the inhibitory population as well. C, D: Same as Figure 4.4C, D except top-down external input was provided to the inhibitory population as well. E, F: Same as Figure 4.9C, D except top-down external input was provided to the inhibitory population as well. 105

4.12 **Mismatch responses are observed with stronger mismatched stimuli.** A, B: Same as Figure 4.5 except the strength of the mismatched input was increased by six-fold. 108

4.13 **Mismatch responses are not observed when more populations are considered** Same as Figure 4.5 except more populations were added. Connections between populations are not shown for the simplicity of the diagram. 110

5.1 **Training fixed point from a recurrent neural network A**: Diagram of a recurrent neural network (RNN) with static input, *i.e.,* regression *or* image data trained to produce fixed point firing rate output. **B)**: Eigenvalues of connectivity matrix, W. The unit circle is shown in blue. Stability requires that the eigenvalues of W have a real part less than 1. **C**: Rectified linear function used as an "f-I" curve, where $G = Id$ in Eq.2.22. **D**: Fixed point firing rates using rectified linear simulation for 1000 ms in **C**. The fixed point dynamic described 6 randomly selected neurons of the RNN in **A** with a size of $N = 200$, neurons connectivity are under the stability in **B**, the external input is generated from $N-$dimensional linear regression synthetic data with some noise. 119

5.2 **Gradient-based learning of a fixed point firing rate in a one-dimensional linear regression task. A)** A simple linear regression task to find the best-fit line (orange) from a set of 500 randomly generated data points (blue). Model output, r (orange line), is defined by solutions to the fixed point equation $r = wr + x$ where x is the input and w is the model parameter. **B)** The cost as a function of w. The cost is convex and has a minimum value at w^* (red star), but the slope of the landscape is very different on each side of the minimum (*i.e.* $w_{left} = -2$, a blue dot on the left side *vs.* $w_{right} = 0$, black dot on the right side). **C,D)** Gradient descent on w starting from initial values w_{left} and w_{right}, respectively with various learning rates. 133

5.3 **Learning a fixed point firing rates in a one-dimensional linear regression task under a learning rule derived from a re-parameterized model. A)** The new parameter $a = F(w)$ as a function of w. **B)** The cost as a function of a is quadratic. The red star shows the minimum and the blue and black circles show the same initial states, as in Figure 5.2A, mapped to the new parameter space. **C,D)** Same as Figure 5.2C,D except using the learning rule from Eq. (5.35). This is equivalent to using gradient descent on a (*i.e.*, Eq. (5.33)) on the cost landscape in panel A. 136

5.4 **Learning a fixed point firing rates in a one-dimensional linear regression task under a learning rule derived from the linearized re-parameterized model. A, B)** Same as Figure 5.3C, D except using the learning rule from Eq. (5.36). 139

5.5 **Visualizing the cost landscape in a higher dimensional linear regression task. A)** The cost function $J(W(t))$ as a function of t from Eq. (5.41). This represents the cost evaluated along five random line segments in $\mathcal{R}^{N\times N}$, each passing through W^* at $t = 0$. Two blue dash lines show the stability boundary, $|t| = 0.346$. The vertical axis is cut off at $J = 1000$ to better visualize the curves. Blue and black circles show stable and unstable initial conditions used later for learning. **B)** The cost function $J(W(t_1, t_2))$ from Eq. (5.42). This represents the cost evaluated on a randomly oriented square with a center at W^*. The color axis is cut off at $J = 1000$. 142

5.6 **Performance of three different learning rules for a linear regression problem in many dimensions.** Three different learning rules: ΔW_1^i (top row; A, B), ΔW_2^i (middle row; C, D), and ΔW_3^i (bottom row; E, F) applied to a linear regression problem in $N = 200$ dimensions with $m = 100$ data points, so the system is over-parameterized ($N > m$). In the left column (A, C, E) the initial W was drawn from within the stability region (spectral radius < 1; blue dot in Figure 5.5A). In the right column (B, D, F) the initial W was drawn from outside of the stability region (spectral radius > 1; black dot in Figure 5.5A). 145

5.7 **Angles and correlations between weight updates for different learning rules. A)** Angle (θ_{12}) between the weight updates for the gradient-based and re-parameterized learning rules. **B)** Correlation coefficient (ρ_{12}) between the weight updates for the gradient-based (ΔW_1) and re-parameterized (ΔW_2) learning rules. **C-F)** Same as A and B, but for all other pairs of learning rules. 149

5.8 **Performance of three different learning rules for a linear regression problem in many dimensions using an under-parameterized model.** Same as Figure 5.6, but for an under-parameterized model with $m = 500 > N = 200$. The dashed red line indicates the true minimum cost, $J(W^*)$. 150

5.9 **Training fixed point from RNNs for a categorical task A**: Diagram of a RNN with MNIST-hand written digit inputs. **B)**: Eigenvalues of the connectivity matrix, W in the network, same as in Figure5.1B. **C**: Sigmoidal function used as an "f-I" curve, $f = tanh(I)$, where $G \neq Id$ in Eq.2.22. **D**: Same as in Figure5.1D, except the input is static image data. 152

5.10 **Supervised linear learning of the fixed point firing rates on an over-parameterized categorical task. A)** Three different learning rules: ΔW_1^i in Eq. (5.55) (top row; A, B), ΔW_2^i in Eq. (5.54) (middle row; C, D), and ΔW_3^i in Eq. (5.56) (bottom row; E, F) applied to a categorical MNIST benchmark problem in a linear network of $N = 200$ dimensions with $m = 100$ data points, so the system is over-parameterized ($N > m$). The cost $J(W)$ is computed from the entropy loss in Eq. (5.52) across 500 iterations. The left column (A, C, E) is the training loss. Right Column (B, D, F)is the training error. 155

5.11 **Supervised linear learning of the fixed point firing rates on the under-parameterized categorical task.** Same as in Figure 5.10, except this is an under-parameterized system. 156

5.12 **Supervised learning of fixed point firing rates on a categorical task using a rectified linear activation function on a support set.** Same as in Fig. 5.10, except using the rectified linear activation from Eq. (5.57). Specifically, **A,B)** applied the learning rule from Eq. (5.58). **C,D)** used the learning rule from Eq. (5.60). **E,F)** performs the learning update from Eq. (5.61) 160

5.13 **Same as in Fig. 5.12 except this is an under-parameterized system.** With $m = 500$. 161

5.14 **Supervised learning of fixed point firing rates on a categorical task using a sigmoidal activation function.** Same as in Fig. 5.10 and Fig. 5.12, except using the tanh activation from Eq. (5.62). Specifically, **A,B)** applied the learning rule from Eq. (5.64). **C,D)** used the learning rule from Eq. (5.64). **E,F)** performs the learning update from Eq. (5.65). 164

5.15 **Same as in Fig. 5.14 except this is an under-parameterized system.** With $m = 500$. 165

TABLES

1.1 Thesis Overview 8

5.1 Runtime Learning Rule Comparison in Higher Dimensional Linear Least Squares Problem 148

5.2 Learning Rule Runtime and Error Comparison of a Categorical Task via Linear Learning 157

5.3 Learning Rule Runtime and Error Comparison of a Categorical Task through ReLu Activation with Support Sets 159

5.4 Learning Rule Runtime and Error Comparison of a Categorical Task through Tanh Activation 163

ACKNOWLEDGMENTS

I would like to first thank my advisor, Dr. Robert Rosenbaum, for showing his passion for the field of computational neuroscience. His guidance in connecting interdisciplinary subjects led me to a new direction in machine learning, which embarked on my journey of discovery. Thanks to his encouragement and support to actively seek opportunities for me. I presented my work at conferences and took several advanced courses through Neromatch Academy for two summers. I would also like to thank him for the computer that he invested in and has accompanied me every day. I am very grateful to receive tremendous support from Dr. Rosenbaum, even during the COVID-19 pandemic, our research discussions never stopped. In addition, I am fortunate to have his wisdom and advice for my career development and job applications.

I would like to thank the National Science Foundation, NeuroNex, and Air Force Office of Scientific Research to fund my research within Dr. Rosenbaum's Neural Dynamics and Computing group at the University of Notre Dame. I would also like to further thank my lab mates Dr. Cody Baker, Dr. Ryan Pyle, and Dr. Christopher Ebsch for taking the lead and introducing me to many of our interesting group discussions.

I would like to thank the graduate school and department of ACMS at Notre Dame for selecting me as the Arthur J. Schmitt Presidential Leadership Fellowship recipient for 5 years. I am grateful to serve as a Schmitt board member and get involved in organizing many campus events to give back to our community. It has been such a rewarding and enlightening experience.

I would like to thank many professors at Notre Dame for generously offering their time and support in my graduate career by providing great resources, especially Dr. Allan Lindsay, Dr. Daniele Schiavazzi, Dr. Jonathan Hauenstein. They have helped me gain the applied and computational knowledge in mathematics and statistics, and continuously supported several of my grant applications. I would also like to thank Dr. Francis Bilson Darku, Dr. Huy Huynh, Dr. Jennifer Waddell, and Dr. Molly Walsh for their extraordinary support of my teaching practice at Mendoza college of business for three years.

I would like to thank the incredible ACMS administrative staff, Kathy Phillips, Melissa Ornat, and Nancy Hermanof for their amazing assistance in many aspects. Their support have warmed my heart throughout the winter time here!

I would like to thank my colleagues, especially Adrian Navarro Hernandez, Bingyue Su, Diana Morales, Jiacheng Zhang, Yihao Fang, and Yu Wang, at Notre Dame for their companionship. We have taken many challenging classes and pulled all-nighters together side-by-side preparing for our qualifying exams.

I would like to thank everyone who has talked to me during my undergraduate and graduate education career, including my students, classmates, friends, and internship managers. Those conversations we had have inspired my life experiences.

Finally, I would like to thank my lifetime friends, Grace Chen, Carole Sun, Yuan Lam, Ana Wright, Dennis Wu, Eli Selkin, Eric Hranchak, and Prof. Ribeiro. I always appreciate everything they share and all they have done to support my higher educational journey.

SYMBOLS

$a.k.a.$	also known as
ms	milli-seconds
mV	milli-Volts
Hz	Herz, reciprocal of seconds
$O.D.E.$	Ordinary Differential Equations
$L(\cdot)$	Loss, a function maps an event or more variables onto a real value
$J(\cdot)$	Cost, an average of the loss over the entire samples

CHAPTER 1

INTRODUCTION

This is the opening of our brain-modeling Odyssey!

The **brain** is a mass of nerve tissue that serves as a control system, the center of the nervous system, and a powerful computing unit of an organism. Body movements, memory, feelings, information processing, and problem-solving are all directly related to the brain's abilities of coordination and functioning. As a complex organ, the human brain is made up of about 100 billion neurons. **Neurons** are the special cells in the brain and function like the basic working units. They communicate with each other in the brain regions (a.k.a. lobes, see Figure 1.1), and each region is associated with certain operations such as emotions, speech, thinking, and learning. Many of these cognition processes are triggered in the cortical area where neurons are densely connected (Figure 1.1b). Understanding how the brain integrates neurons within their domain regions and present operations is a fundamental and challenging question in computational neuroscience.

1.1 A Brief History of A Brain-inspired Machine Learning Architecture

The structure and mechanism of how brains perform computations have inspired scientists like Warren McCulloch and Walter Pitts, who built the first **artificial neural networks (ANNs)** back in 1943 [89]. An ANN is a special type of computational model in machine learning that uses neuron-like interconnected nodes to flow information. As a computing system, ANNs have made great strides in their ability to learn from past experience and their ability to solve difficult learning tasks

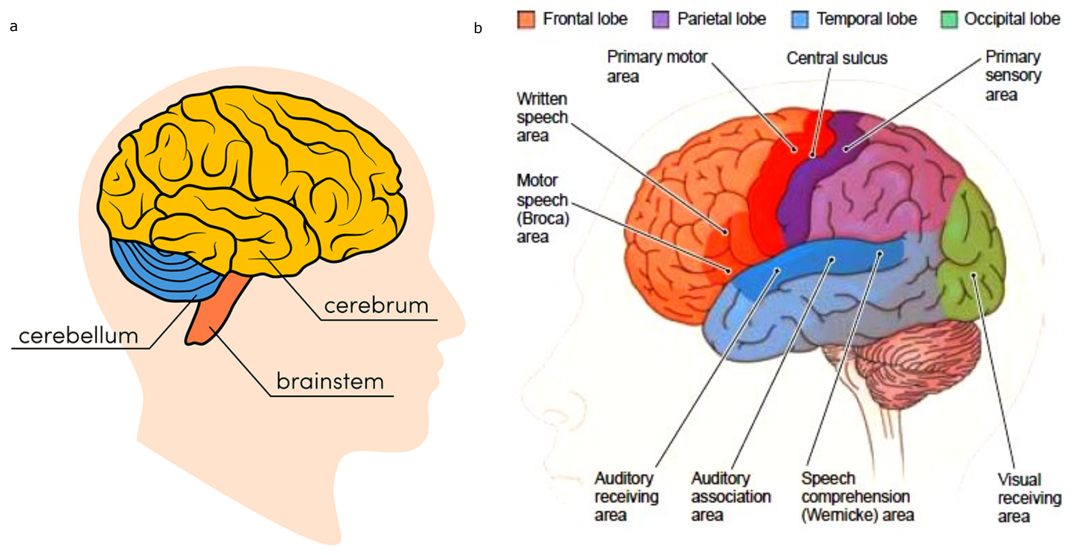

Figure 1.1. **Brain regions.** a: Brain division. b: Cortical regions with areas that are associated with several specific cognition tasks. Image is taken from Google Search.

over the last decade. From speech recognition, and machine translation, to social network filtering, these applications have become powerful tools and infiltrated into our life experience in every aspect. Although the original goal of designing neural networks was to solve problems in the same way that a biological brain would do, there are stark differences between biological and machine learning, and ANNs remain inferior to the brain at general intelligence and generalizing across tasks. Moreover, biologically realistic computational models cannot perform well on non-trivial tasks.

The advances in artificial intelligence in recent years have motivated many neuroscientists to seek ways in which ANNs can be used to better understand the brain and likewise, whether advances in neuroscience can inform the developments of improved ANNs. While ANNs share some of their basic building blocks with cortical neuronal networks in the brain, they differ in some fundamental ways. Although there has been some work in linking biological and artificial neural networks, current

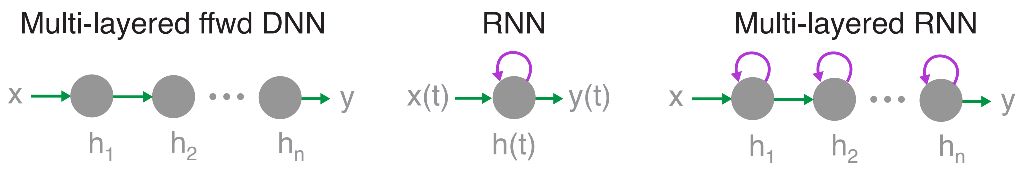

Figure 1.2. **Neural network architecture in machine learning** Left: multi-layered feedforward networks (a.k.s DNN) are used to learn static representations $y = F(x)$. Middle: RNNs are used to learn functions between time series, $y(t) = \mathcal{F}[x](t)$. Right: Multi-layered RNNs that is cortical circuits alike and learn static functions as well as functions between time series.

research that uses ANNs to model cortical circuits utilize either single-layer recurrent neural networks (RNNs, Figure 1.2 middle) or multi-layered feedforward deep neural networks (ffwd DNNs, Figure 1.2 left), whereas biological neuronal networks in the cerebral cortex are multi-layered and recurrent (Figure 1.2 right). Hence, one major direction of establishing direct, one-to-one analogues between artificial and biological neuronal networks is through the recurrent network structure.

1.2 Recurrent Neural Networks (RNNs)

A neural network structure that not only processes the information in a forward manner (see in Figure 1.2Left) but also allows the signal to loop back into the network itself recurrently is called a **recurrent neural network** (a.k.s. RNN, see in Figure 1.2Middle and Right). RNNs are both widely used in machine learning and in computational neuroscience contexts.

1.2.1 Artificial Recurrent Neural Networks (ARNNs)

In machine learning, RNNs are a subclass of ANNs, and they are a type of neural network structure where the output also feeds as input going into the same unit recurrently (Figure 1.3). Unlike ANNs where the input and output are independent,

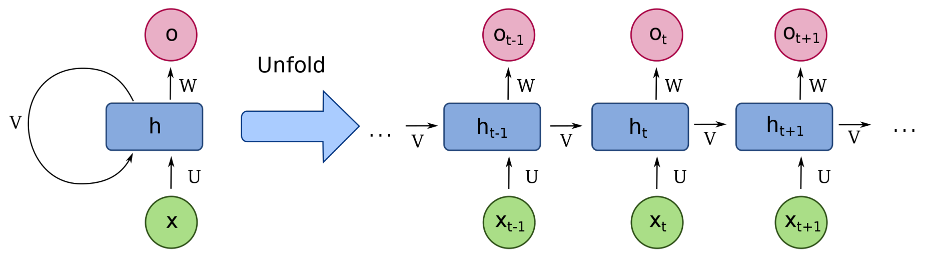

Figure 1.3. **A single-unit RNN diagram**. Left: a compressed unit. Right: the unfolded version of the compressed unit. From bottom to top: time series input X_t, hidden layer h_t, and output O_t. U, V, and W are the connectivity weights of the network. Image is taken from Wikipedia RNN Search.

the "recurrent" motion breaks this independence and forces the input and output to become dependent in RNNs. The most simple neural network structure is the one-layer ANN (imagine there is no recurrent V connection in Figure 1.3 left), and likewise, the simplest RNN has one single hidden layer with a feedback connection architecture like in Figure 1.3 left *or* in Figure 1.2 middle. The "recurrent" feature of RNNs can be viewed as a direct cyclic graph, which allows the output from some nodes to influence back to themselves through subsequent of steps (a.k.a. hidden layers). This ability of information revisiting through a sequence of hidden layers gives RNNs the "memory" characteristic. Hence, RNNs can use these internal hidden states to process and learn sequential input like text streams, audio *or* video clips, and time series data (Figure 1.3 unfold part on the right). This learning through time and dependent input neural network structure has demonstrated its advantages in many engineering tasks such as language modeling, text generating, speech and image recognition, machine translation, and all kinds of detection.

Apart from their ability for solving difficult tasks, RNNs in general are *not* considered as a biologically realistic model. For example, in this architecture, neurons do not have any real biological meaning since unlike biological neurons, they are *neither* excitatory *nor* inhibitory. These neurons are artificial and they are more like nodes in the graph for a symbolic representation. The naming convention is inherited from the architecture of interconnected neurons within a brain region. We refer to this type of RNN as artificial RNNs (ARNNs, to distinguish it from a more biologically realistic counterpart, BRNNs in 1.2.2), and we often use ARNNs in related to more practical and technological applications.

1.2.2 Biological Recurrent Neural Networks (BRNNs)

In computational neuroscience, the brain can be divided into regions like in Figure 1.1. Neurons are the basic computational processing elements within these regions, hence they have their real biological meaning. As a result, they can be classified into distinct neuron types and release electrical and chemical molecules (a.k.a. neurotransmitters) to affect others by *either* exciting *or* inhibiting adjacent neurons. However, the cortical region of the brain is not simply arranged as confined layers of neurons, so the recurrent connections in machine learning are not the same as the notion of recurrence in the neuroscience literature. It is akin to the *lateral* connections, meaning the interconnected neurons are in a localized region. In addition to the feed-forward and lateral connections, the feedback signal is ubiquitous in the brain, whereas it's often absent in ARNNs (see Figure 1.4 as an example and compare it to Figure 1.3 in Section 1.2.1). One reason may be related to the training difficulties because ARNNs would need new learning rules other than gradient descent back-propagation in order to obtain the same functionalities of feedback connections that appear in the biological recurrent neuronal networks (BRNNs). In contrast, the neuron responses in brain dynamics are often described by their firing rates using ordinary differential equations, where some biological constraints are incorporated. For example, the neuron synaptic strength constraint is based on the properties of

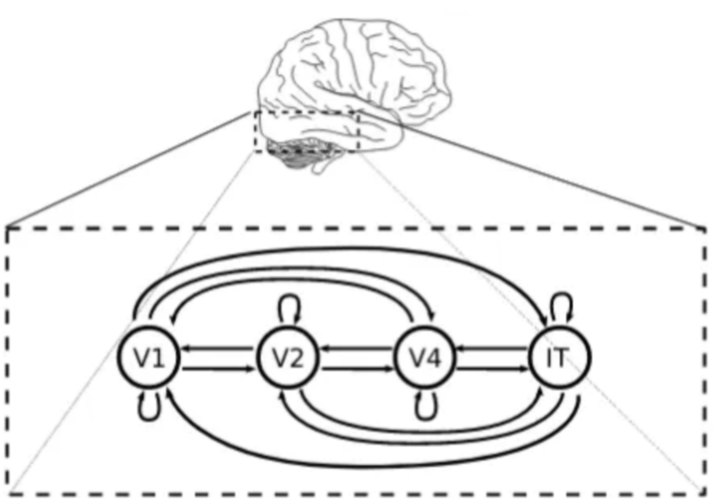

Figure 1.4. **A 4-layered biological RNN.** A biological RNN architecture to model the visual cortex from Liao and Poggio's experiments. The network contains feed-forward connection (left-to-right), feed-back connections (right-to-left) and lateral connections (looping back to same area; synonymous with recurrent connections in ARNNs terminology). [54]

neurotransmitters, so models in BRNNs can provide a more realistic interpretation.

Although ARNNs and BRNNs share some structural similarities, computational processing and training are very different. In ARNNs, each unit works synchronously to process sequential data through gradient-based learning methods, whereas, in BRNNs, neurons send their signals asynchronously. As a result, models in BRNNs can handle information in a distributive and independent parallel way. Also, the learning method in the human brain would be quantitatively more diverse than the standard gradient descent back-propagation methods in ARNNs. Therefore, models in BRNNs can perform more complex functions such as memory storing, winner-take-all decision-making, and even for the learning of new tasks. Understanding and simulating the same functionalities of the brain through BRNNs becomes a crucial step in the development of ARNNs.

1.3 Overview and Contributions

In this thesis, I will address various differences and similarities between artificial and biological neural networks in my research through model development, training, and performing learning tasks in RNNs. I will show the connection between computational neuroscience and machine learning through my work in neuron response modeling, neuron plasticity, and the fixed point of recurrent dynamics learning in RNNs. I will start with some basics in neural computations and biological spiking models for a single neuron to a population of neurons; then introduce the mean-field rate approximation and balanced network theory that are commonly used to model cortical circuits. I will cover my work with theorems that characterize features of cortical networks and can directly link to some machine learning activations. I will further discuss my work on the prediction errors in unstructured neuronal networks through homeostatic plasticity computations. Finally, I will present the fixed point of recurrent dynamics learning performance in the BRNN learning algorithms that I developed.

The topics included in this thesis are shown in table. 1.1. Chapter 2 provides the necessary definition, biological background knowledge, and mathematical modeling. Chapter 3 discusses the traditional balanced mean-field theory in detail and my work toward its new extension. Chapter 4 describes the importance of network structure for the success of learning predictive coding for non-trivial tasks with homeostatic plasticity. Chapter 5 introduces a re-parameterized learning algorithm for training the fixed points of RNNs and other alternative forms, which can be robust and computationally efficient in compared to the traditional gradient-based learning methods. Finally, Chapter 6 gives a summary and discussion of my future work directions.

TABLE 1.1

THESIS OVERVIEW

Chapter	Contributions
3	Expand the traditional balanced network by introducing a "semi-balanced" state and draw connections to machine learning activations.
4	Discuss whether or not the homeostatic plasticity can learn to compute prediction errors in an unstructured network.
5	Develop two new BRNN learning algorithms for the fixed point of recurrent dynamics and demonstrate their learning capacities.

CHAPTER 2

BIOLOGICAL BASICS AND MATHEMATICAL MODELING

In this chapter, I will introduce the biological background of the brain and its working element – the neuron, with a focus on the modeling of a single neuron to multiple neurons, their communications through synapses, and responses within a network.

As the main focus of this dissertation is from the perspective of mathematics and computations for neuron dynamics, a very brief review of the brain structure and how it is related to neurons' activities is sufficient. I will leave exploring the vast biological complexity of the brain on the side.

2.1 Biological Neurons

Neurons are the nerve cells living in the brain. They are the information carriers that receive, process, and transmit electrical and chemical signals to other cells. There are different types of neurons that vary in shape and functionalities. Depending on their roles and locations (see Fig. 2.1a), we can have neurons that are classified as unipolar, bipolar, or multipolar. We can also have neurons that are pyramidal and Purkinje-like. However, they all share a common structure from the top down contain a cell body (a.k.a. **soma**), an **axon**, and **dendrites** (see Fig. 2.1b). The cell body contains the nucleus and specialized organelles where the genetic information is located. Like any other cells in a body, soma is enclosed by a **membrane** that can selectively stop the surroundings while passing the others. A neuron also contains a tail-like axon where it connects the cell body to other neurons by a white fatty

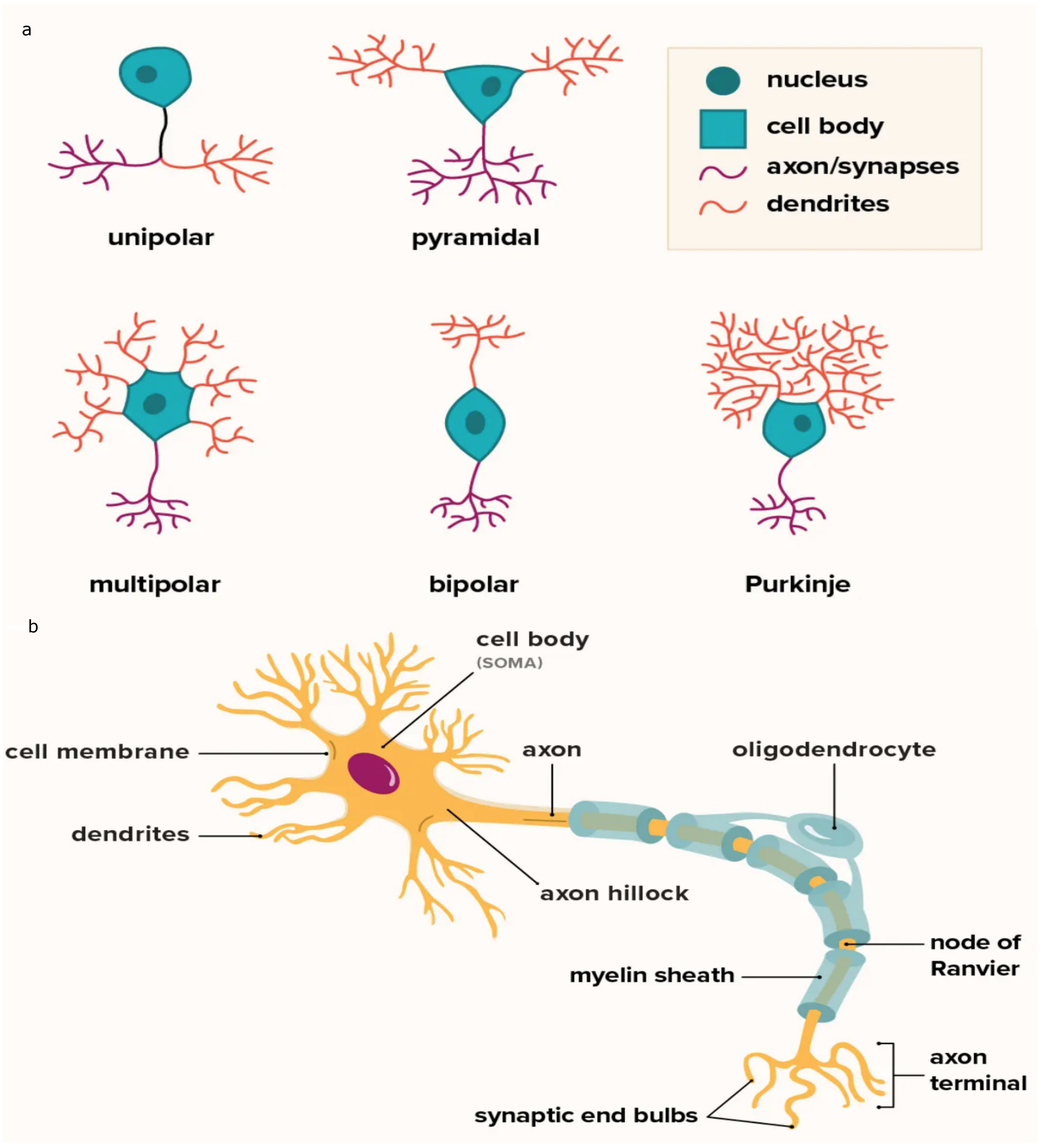

Figure 2.1. **Neuron shapes and its structure.** a: Illustration of neuron morphologies vary in size, shape, and location. b: The basic structure of a neuron. Image is taken from Healthline Media: https://www.healthline.com/

substance called myelin (white matter in the cortical region is “white” because of the large quantities of myelinated axons). Myelin acts like an insulated coating which is similar to the plastic rubber around the wire cord of a computer charger. The fatty layer can protect the electrical energy from leaking, so it helps the axon send electrical signals. Dendrites are responsible for receiving and detecting spikes from other neurons. They are the information collectors and processors, so a neuron may have more than one dendrite, like the branches of a tree.

Neurons process the information by sending electrical impulses and chemical signals via **neurotransmitters** throughout the brain. When a neuron receives electrical excitation from other transmitting neurons, it causes the transmitting neuron to open the neurotransmitter vesicle and release chemicals (a.k.a. neurotransmitters). Information is transported along with the neurotransmitters traveling from the axon of the transmitting neuron, across the tiny gap (a.k.a. **synaptic** cleft) between the terminal to the dendrite of receiving neuron (Fig. 2.2). This process allows information exchange and enables us to feel, think and move around.

There are some negative and positively charged ions in the interior and exterior sides of the neuron membrane. Resting neurons are negatively charged from the inside out, and this difference between the electrical potentials across neuron membrane is called **membrane potential**, often denoted as V. At a stable stage, the membrane potential is around -70mV. Although the membrane itself is a blocker, the membrane potential is controlled by the ions flowing, and this is further dependent on the **ion channels** (a.k.a. ion pumps) located in the membrane. When neurons receive stimulus from one end by an electrical or neurotransmitter activity, it also creates a voltage change across their membrane. The release of neurotransmitters may result in a positive increase (or negative decrease) in the electrical charge of a receiving neuron, hence causing the membrane to be depolarized (or hyperpolarized). In general, neurotransmitters follow Dale’s Law such that a neuron releases the same

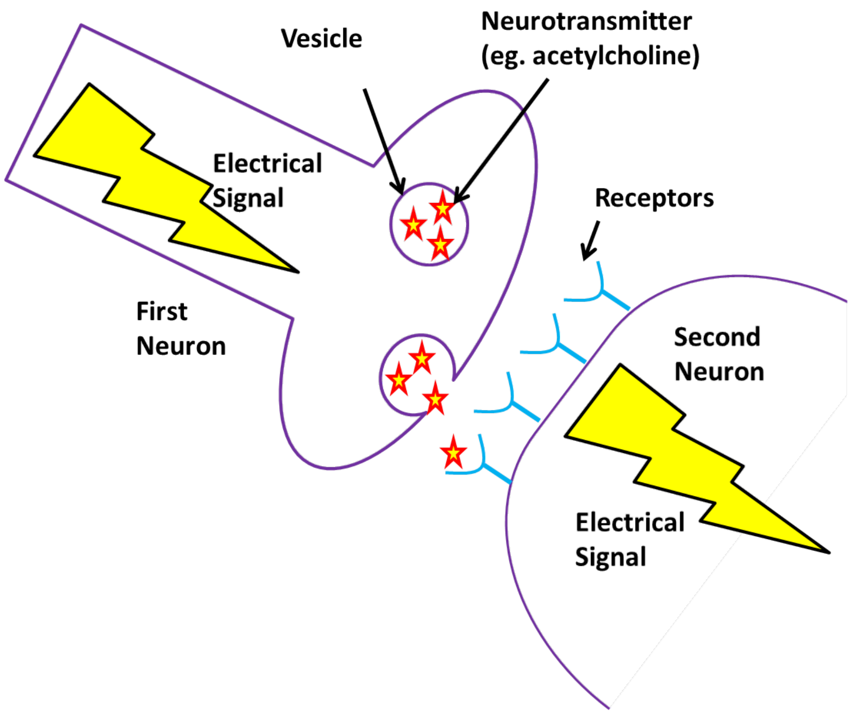

Figure 2.2. **Neuron communication diagram**. First Neuron (a.k.a. transmitting *or* presynaptic neuron). Second Neuron (a.k.a receiving *or* post-synaptic neuron). The image is taken from a Google Image search.

neurotransmitter (*either* excitatory *or* inhibitory signal, see section 2.2.3) to the receiving neurons.

Neurons communicate electronically through a synapse, an **action potential** (a.k.a. spike) occurs when the incoming excitatory signals are sufficiently large and result in the receiving neuron depolarization. After an action potential takes place, it quickly moves to a resting stage because the ion channel cannot open again immediately. This is called **refractory period**, and it usually takes about 1 $\sim$ms, which allows enough time to initiate another action potential, and hyper-polarization may happen during this time period. After a spike, the information is processed as the electrical signal has passed through from the transmitting neuron to the receiving neuron and reached its axon, where the vesicle is ready to release its neurotransmitters to another targeting neuron. Hence, the term transmitting and receiving neurons are relative to a particular synapse, since most neurons are both **pre-synaptic** and **post-synaptic**.

Note that not every increase of electrical charge will result in an action potential since the action potential is the result of very rapid changes in voltage when crossing the neuron membrane, so the threshold is approximately -55mV. Also, stronger stimuli will not result in the change of action potential in magnitude; instead, it will result in multiple action potentials and affect the frequency, the sequence of a neuron's action potentials is called the **spike train**. We often use mathematical models to study the neuron spiking activities, either a model to depict neuron membrane dynamics, or a model to describe the frequency.

2.2 The Modeling Development of A Single Neuron

From a biological perspective, neurons communicate via synapses. This process can trigger an action potential if the membrane potential voltage exceeds the threshold level, and there is a refractory period after the spike where the neuron cannot initiate another spike. Despite the biological complexity of a single neuron, we can analyze the computational characteristics of a neuron since the information is encoded as spiking activities through time and frequency. This provides another scope to understand brain dynamics, and from this perspective, we can also say that neurons communicate primarily through spikes. To understand how the brain functions in a more flexible and general sense, we need to use our mathematical modeling tool for neuron dynamics.

2.2.1 Leaky Integrator Model

In 1907, Lapicque proposed a model to monitor membrane potential over time. The goal is to measure the current, $I(t)$, flow across the neuron membrane is related to its voltage difference through a simple linear ordinary differential equation (ODE) trajectory

$$I(t) = C_m \frac{dV}{dt}. \tag{2.1}$$

Here, the neuron membrane is acting like a leaky capacitor, C_m (it is "leaky" because the membrane is not a perfect insulator, the electrical charge will slowly leak). When there is no current flow, the cell membrane is in a resting state (a.k.a. **reversal potential**, denoted as E_L). Typically in cortical neurons, $E_L \approx -70mV$. We can approximate the leaky current around reversal potential by

$$I_L = -g_L(V - E_L) \tag{2.2}$$

where g_L is the leak conductor that quantifies the resistant level of the membrane from iron passing. Under the stimulation of an external current $I_x(t)$, we can combine Eq. 2.1 in an inward direction where the current is positive and Eq. 2.2 together to obtain the **Leaky Integrator** model[27, 32]

$$\tau_m \frac{dV}{dt} = -(V - E_L) + I_x(t) \tag{2.3}$$

where $\tau_m = \frac{C_m}{g_L}$ is a timescale constant for the membrane potential, and usually it is around $5 \sim 20$ms. The analytical solution of this model contains an exponential decay term, and this corresponds to the membrane potential dynamics eventually back to the resting state. For a simple example, if we have a constant input (*i.e.,* $I_x = I_0$), and let V_0 be the initial state, then the solution becomes $V(t) = (V_0 - E_0 - I_0)e^{-t/\tau_m} + (E_L + I_0)$. Although it depicts the membrane dynamics, it does not capture action potentials. As mentioned in Section. 2.1, a spike is initiated if $V > -55$mV threshold.

2.2.2 Exponential Integrate-and-Fire (EIF) Model

To improve the Leaky Integrator model in Section. 2.2.1 and include action potentials, we can manually "record" the time, and send the membrane potential back to the resting state V_{re}. Besides capturing the motion of action potentials, there is another important part missing. Recall from section 2.1 that the spike is initiated when there is enough electrical charge from the ions flow, and this is controlled by the opening and closing of ion channels. The spiking activities only happen if the membrane potential exceeds the threshold V_{th}. This part can be achieved by adding an exponential term on top of the Leaky Integrator model from Section. 2.2.1. Hence it is called **Exponential Integrate-and-Fire(EIF)** model.

$$\begin{aligned} &\tau_m \frac{dV}{dt} = -(V - E_L) + De^{\frac{V-V_T}{D}} + I_x(t) \\ &\text{if } V(t) > V_{th}, \text{record a spike at time t; then reset } V(t) \to V_{re}. \end{aligned} \tag{2.4}$$

The exponential term incorporates the influx of ion (specifically, sodium Na^+) channels. V_T acts like a sub-threshold, so $V_T < V_{th}$. This is to ensure the opening of sodium channels prior to the action potential, and this soft threshold is around -55 mV. Again, $D \approx 1 - 5$ mV is the timescale factor, and note that when $D \to 0$, the EIF model is equivalent to the Leaky Integrator model in Eq. 2.3.

Section. 2.2.1 to 2.2.2 can be summarized by putting all the right-hand side of the equations (Eq. 2.3 and Eq. 2.4) as a function of membrane potential,

$$\frac{dV}{dt} = f(V) \tag{2.5}$$

where

$$f(V) = \frac{-(V - E_L) + De^{\frac{V-V_T}{D}} + I_x(t)}{\tau_m}. \tag{2.6}$$

For simplicity, if we have a constant input $I_x(t) = I_0$, then we can analyze the action

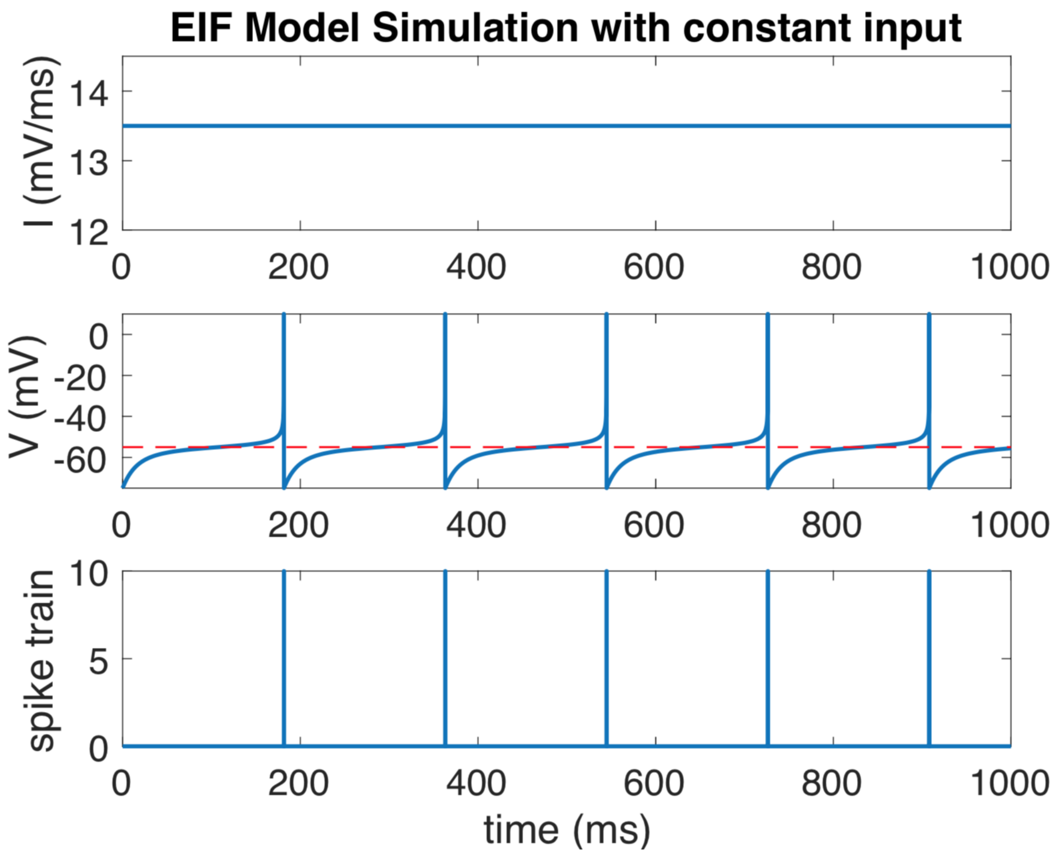

Figure 2.3. **EIF model simulation**. Top: a constant external input injection. Middle: the membrane potential dynamics. Bottom: a spike train simulation is to describe the middle plot.

potential via the **fixed points** (meaning no change in time, hence $\frac{dV}{dt} = 0$) solution from Eq. 2.5. When D is small, there are two fixed points solutions, near $V = E_L + I_0$ is the stable one and near $V = V_T$ is the unstable fixed point. That is to say if the membrane potential is weak and not reaching the sub-threshold ($V < V_T$), then there is no spike since it will converge to the stable fixed point at $V = E_L + I_0$. This situation corresponds to the Leaky Integrator model that has no spike. However, if the membrane potential exceeds the sub-threshold ($V > V_T$), then it will continue to increase until it reaches the threshold V_{th}, hence producing a spike.

An increase in external current will push two fixed points towards each other until they overlap, then eventually disappear since $f(V) > 0$ as $I_0 \to \infty$. That is to say, the membrane potential will exceed the threshold, then follow with a sequence of spikes as discussed in Section 2.1, also see in Figure 2.3.

2.2.3 Synapses-Driven Model

Previously, the neuron spiking model is driven by an external stimulus as a form of an electrical current. Neurons communicate through synapses, like the interactions between two neurons presented in Figure 2.1. This indicates that a neuron spiking model can be also driven by a stimulus current that is coming from another neuron, and this is called the **synapse-driven** model. When a pre-synaptic neuron first spikes (as a result of other driven forces), then its ion channel opens and releases neurotransmitters, these chemicals cross the synaptic cleft and reach the post-synaptic neuron. This process also brings a current to the post-synaptic neuron, so we called it a **post-synaptic current (PSC)**. PSC affects the membrane potential of the post-synaptic neuron. Hence, every pre-synaptic neuron spike will result in a **post-synaptic potential (PSP)** response. PSP gives a measurement of the connectivity strength, which is associated with the pre-synaptic neuron.

Based on the results of synapse current, we can classify neurons as excitatory and inhibitory. **Excitatory** neurons increase the likelihood that excites the post-synaptic neuron to spike, whereas **inhibitory** neurons reduce the likelihood of spiking. There are 80% expiatory neurons and 20% inhibitory neurons in the cortex. According to Dale's Law, all the post-synaptic neurons will inherit the same type of synapse, so either excitatory or inhibitory. To model a current-based synaptic model, we need to describe the synapse current as a sum of PSC (*either* EPSCs *or* IPSCs)

$$I_a(t) = J_a \sum_j \alpha_a(t - t_j^a). \tag{2.7}$$

J_a is the **synaptic weight** scalar which measures the strength of a synaptic current ($J_e > 0$ and $J_i < 0$). $\alpha_a(t)$ is the **PSC waveform**. Since there is no post-synaptic response prior to the pre-synaptic spike, so one can assume that $\alpha_a(t) = 0$ when $t < 0$, and $\int_0^\infty \alpha_a(t)dt = 1$ when $t > 0$. We have a couple of options here to model the waveform. A commonly used function to capture the smoothness "wave" (as a production result from the chemical releasing components of the neurotransmitters) is given by the exponential decay form described as

$$\alpha_a(t) = \frac{1}{\tau_a} e^{-\frac{t}{\tau_a}} H(t) \tag{2.8}$$

where $a = e, i$. $H(t)$ is the Heaviside function such that $H(t) = 1$ whenever $t > 0$, and $H(t) = 0$ otherwise. To model the membrane potential of a post-synaptic neuron in Figure 2.4, we can develop a synapse-current-based leaky integrator model, and it is driven by two external neurons as

$$\tau_m \frac{dV}{dt} = -(V - E_L) + I_{syn}(t) \tag{2.9}$$

where $I_{syn}(t) = I_e(t) + I_i(t)$. For simplicity, one can model the synaptic current input as a point process instead of a smooth "wave" form. Dirac delta function becomes

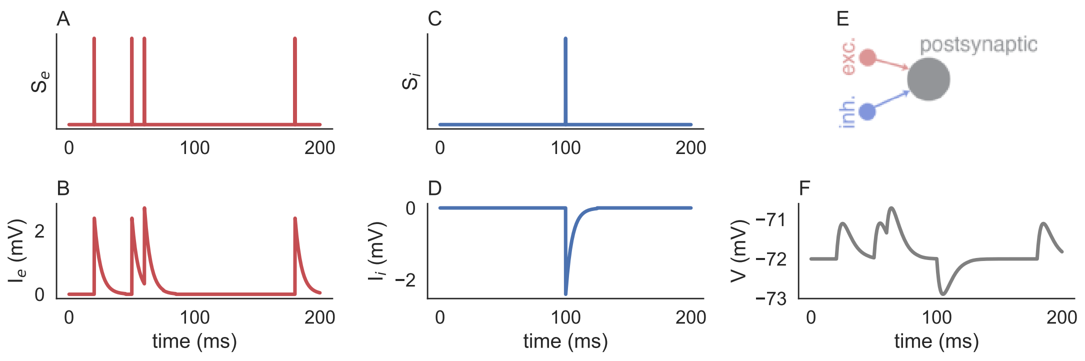

Figure 2.4. **Leaky integrator Model driven by two pre-synaptic neurons**. A: excitatory spike train. B: excitatory synaptic current. C: same as A and B, except for the inhibitory neuron. E: synapse diagram with two Pre-synaptic neurons and one post-synaptic neuron. F: the membrane potential dynamic of the post-synaptic neuron.

a natural choice for a form of $\alpha_a(t)$ to model each spike. In this case, we ignore the "wave" shape, and obtain a **spike train**, $S_a(t)$, by adding all these Dirac delta functions at each time point, t_j, as

$$S_a(t) = \sum_j \delta(t - t_j^a). \tag{2.10}$$

Spike train provides a new way to write the Eq. 2.7 in terms of **convolution**, $I_a(t) = J_a(\alpha_a * S_a)(t)$. We can count the number of spikes in another way as $N(t_1, t_2) = \int_{t_1}^{t_2} S_a(t)dt$, and this holds the same results as the usual counting method. Within any time window (t_1, t_2), we can also easily get the frequency per unit in time measurement, and this is called the **firing rate** (in Hz, meaning "number of spikes per milli-seconds"),

$$r = \frac{N(t_1, t_2)}{t_2 - t_1}. \tag{2.11}$$

The study of neuron firing rates gives us opportunities to understand its computational properties (*i.e.,* mean, variance, etc.) as well as its biological characteristics.

The spike train in Figure 2.4A is a **stationary process** where the mean and variance do not change over time. A **Poisson process** is frequently used to model a spike train with this feature. The mean value of the spike count from a Poisson process is well understood through the firing rate,

$$E[N(t_1, t_2)] = \int_{t_1}^{t_2} r(t)dt. \tag{2.12}$$

Since the firing rate does not change, so $r(t) = r_0$, then $E[N(t_1, t_2)] = r_0 * (t_2 - t_1)$.

2.2.4 Mean-Field Theory

Model in Eq. 2.9 is not biologically realistic enough yet since the action potentials of the post-synaptic neuron are not displayed. Moreover, a post-synaptic neuron in reality receives many pre-synaptic neurons, and some are excitatory while others are inhibitory (say N_a for each, $a = e, i$). We can put them into a population and develop a more biologically realistic EIF model as,

$$\begin{aligned}
\tau_m \frac{dV}{dt} &= -(V - E_L) + De^{\frac{V-V_t}{D}} + I_e(t) + I_i(t) \\
\tau_e \frac{dI_e}{dt} &= -I_e + \mathbf{J}^e \cdot \mathbf{S}^e(t) \\
\tau_i \frac{dI_i}{dt} &= -I_i + \mathbf{J}^i \cdot \mathbf{S}^i(t) \\
&\text{if } V(t) > V_{th}, \text{record a spike at time t; then reset } V(t) \to V_{re}.
\end{aligned} \tag{2.13}$$

Note that $\mathbf{J}^a$ and $\mathbf{S}^a$ are a vector of size N_a. Figure 2.5 shows a similar characteristic of the input synaptic neurons in comparison to Figure 2.4. For example, instead of having a spike train for a single neuron, now we can produce roster plots for each neuron group in the whole input neuron population. Moreover, the synapse current, $I_{syn}(t)$, in Eq. 2.9 now describes the total input from each neuron group as $I_{syn}(t) = I_e(t) + I_i(t)$.

When there is a population of neurons sending their synaptic currents, we can perform some statistical analysis of these pre-synaptic neurons such as taking the mean current over each excitatory and inhibitory group,

$$\bar{I}_a = E[\mathbf{J}^a \cdot \mathbf{S}^a] = \sum_{j=1}^{N_a} E[\mathbf{J}_j^a * \mathbf{S}_j^a(t)] = \mathbf{J}^a \cdot \mathbf{r}^a \tag{2.14}$$

where $a = e, i$. The final equality follows from the expectation of the spike train in a Poisson process is just the firing rate itself. If the neuron synaptic connection within each group (*either* excitatory *or* inhibitory) have the same strength, (so for all $k \in a$, $J_k^a = j_a$ and their firing rates are time-constant, hence they do not depend on time, $\mathbf{r}_a(t) = r_a$), then the Eq. 2.14 has a **stationary mean value**, $\bar{I}_a = N_a j_a r_a$ (this value should fluctuate around the time-average input, dash line in Figure 2.5C, E and G, $\bar{I}_a = \lim_{T\to\infty} \frac{1}{T}\int_0^T I_e(t)dt$ where $I_e(t)$ represents a single trial). One can now interpret the input currents as a combination of the mean current with some noises, this is an illustration of **mean-field** theory ("field" is to specify a neuron group), the membrane potential of the post-synaptic neuron can be described as

$$\tau_m \frac{dV}{dt} = -(V - E_L) + De^{\frac{V-V_t}{D}} + \bar{I}_{syn} + \text{noise}. \tag{2.15}$$

Note that the noise term can evoke spiking activities especially when the mean input is not so strong (see the spike around 380ms in Figure 2.5).

2.2.5 f-I Curve

On a detailed level, how does the mean input affect the firing rate of a post-synaptic neuron? Since the stationary mean is related to the population size of the pre-synaptic neurons, their synaptic strength, and their time-constant firing rate. If keeping the incoming neuron size and presynaptic strength unchanged, while increasing the excitatory firing rates in Figure 2.6 from A to B, we see that the post-synaptic

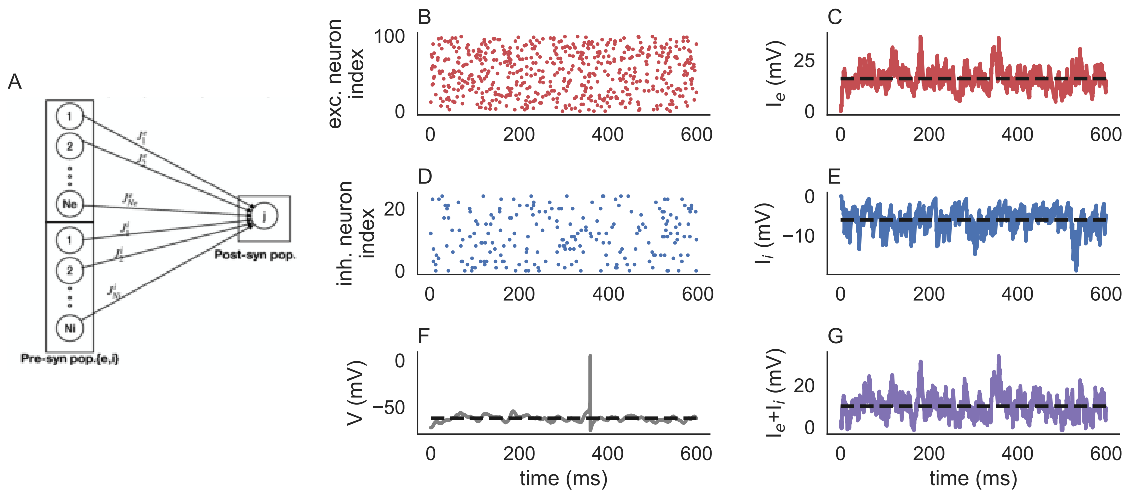

Figure 2.5. **EIF model driven by a population of pre-synaptic neurons**. A: synapse diagram with 100 Pre-synaptic excitatory neurons and 25 inhibitory post-synaptic neurons. B: raster plots of pre-synaptic excitatory neurons. C: mean value of excitatory pre-synaptic neuron current. The dash line is the stationary mean. D, E: same as B and C, except for the inhibitory pre-synaptic neurons. F: the membrane potential dynamic of the post-synaptic neuron. G: the total synaptic input current by summing C and E.

firing rate is also increasing in Figure 2.6C as blue dot to red dot. To be biologically meaningful, neuron firing rates must be non-negative, so we can use a non-negative and non-decreasing function to describe the relationship between the stationary mean current input and the post-synaptic firing rate of a single neuron. This is called the **f-I** curve, and it can be approximated as the following,

$$r \approx f(\bar{I}_{syn}). \tag{2.16}$$

It is often fitted with a **threshold linear** (a.k.a. rectified linear *or Relu* activation in machine learning) function

$$f(\bar{I}_{syn}) = g[\bar{I}_{syn} - \theta]^{+} \tag{2.17}$$

where $g > 0$ is the gain, $\bar{I}_{syn} = N_e j_e r_e + N_i j_i r_i$, and $\theta > 0$ is the threshold such that $f(\bar{I}_{syn}) = 0$ whenever $\bar{I}_{syn} < \theta$, and $f(\bar{I}_{syn}) = g\bar{I}_{syn}$ otherwise. Other choices of the approximations are also possible such as using a hyperbolic tangent fitting or a logistic transformation.

2.3 Modeling of A Network of Neurons

In previous sections, we studied the membrane potential of a single neuron and its external input forms. Models like EIF and its variants are called **spiking models** because they are designed to capture the action potential. We now extend these models to a group of post-synaptic neurons and study their biological and computational properties. These neurons together with their pre-synaptic connections formed a **network**. In graph theory, a network contains nodes and edges. Here, nodes are the post-synaptic neurons, and edges are the information that flows into these neurons, so either from outside of their own population or internal connections.

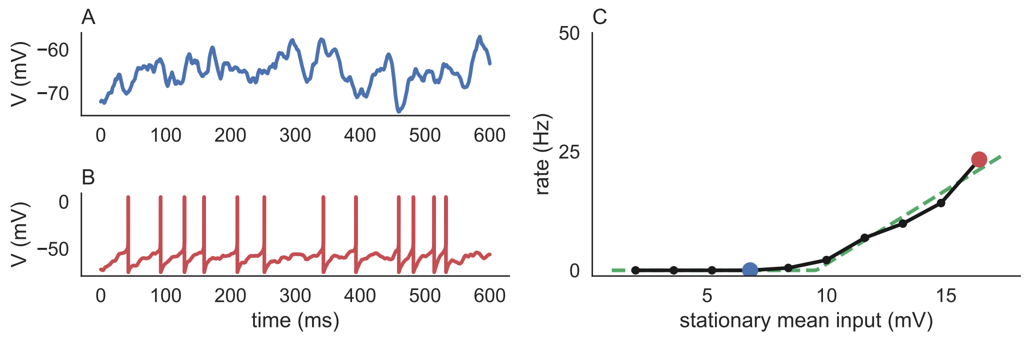

Figure 2.6. **f-I curve for an EIF driven by Poisson synaptic input..** A: Membrane potential of an EIF in a non-spiking regime. B: Membrane potential of an EIF in the noise-driven regime. C: An f-I curve for an EIF. The firing rate was plotted as a function of stationary mean synaptic input. Varying $\bar{I}_{syn}$ through r_e while keeping N_a and J_a unchanged. The blue and red dots correspond to the firing rate from A and B. Dotted green line is the threshold linear fit of data.

2.3.1 Recurrent Network

In a more realistic setting, cortical neurons are **recurrently connected** since they not only connect with neurons outside of the cortex but also receive signals from nearby neurons in the same cortical layer (see Figure 2.7A). This self-supplied synaptic current source can be described in the EIF spiking model as the following,

$$\begin{aligned}
\tau_m \frac{d\mathbf{V}^a}{dt} &= -(\mathbf{V}^a - E_L) + De^{\frac{\mathbf{V}^a - V_t}{D}} + \mathbf{I}^{ae}(t) + \mathbf{I}^{ai}(t) + \mathbf{I}^{ax}(t) \\
\tau_b \frac{d\mathbf{I}^{ab}}{dt} &= -\mathbf{I}^{ab}(t) + \mathbf{J}^{ab}\mathbf{S}^b(t) \\
&\text{if } \mathbf{V}_j(t) > \mathbf{V}_{th}, \text{record a spike at time t; then reset } \mathbf{V}_j(t) \to \mathbf{V}_{re}.
\end{aligned} \tag{2.18}$$

There are three current sources that contribute to the membrane potential dynamics of each neuron in the network (namely the synaptic current, $\mathbf{I}_{syn}$ from excitatory and inhibitory neurons, and the external stimulus $\mathbf{I}_{ax}$ where $a = e, i$). Since the network contains excitatory and inhibitory neurons, the second O.D.E. in Eq. 2.18, $\mathbf{I}^{ab}(t)$ describes the synaptic input from population b to population a, hence it contains six flows where $a = e, i$ and $b = e, i, x$. Taking the expectation for each current dynamic, we obtain the **trial-average** current over the population b, $\bar{I}_{ab} = N_b p_{ab} j_{ab} r_b$. Moreover, from the network receiving side, the total trial-average flow into each population is $\bar{I}_a = \bar{I}_{ae} + \bar{I}_{ai} + \bar{I}_{ax}$.

$\mathbf{J}^{ab}$ is the block-wise synaptic matrix form and it looks like,

$$J = \begin{bmatrix} J^{ee} & J^{ei} \\ J^{ie} & J^{ii} \end{bmatrix}.$$

Similar to earlier when we use J_a in Eq. 2.7 to describe the synaptic weight of a single neuron. Now for a population of post-synaptic neurons in the recurrent network, we can define a $N \times N$ matrix, $\mathbf{J}$, where each entry J_{jk}^{ab} measures the connection strength between the pre-synaptic neuron k in population $b = e, i$ to the post-synaptic neuron j in population $a = e, i$. Specifically, the value of J_{jk}^{ab} depends on the biophysical type of the pre-synaptic neuron, not the distance to the targeting neuron. Hence, we can use the Erdös-Renyi model to obtain

$$J_{jk}^{ab} = \begin{cases} j_{ab} & \text{with probability } p_{ab} \\ 0 & \text{otherwise.} \end{cases}$$

If $J_{jk}^{ab} = 0$, meaning there is no connection from the pre-synaptic neuron k, this is likely to happen with a probability of $1 - p_a$. In reality, most neurons in the cortex are not connected, so we keep p_a small to capture the sparsity of the cortical network structure.

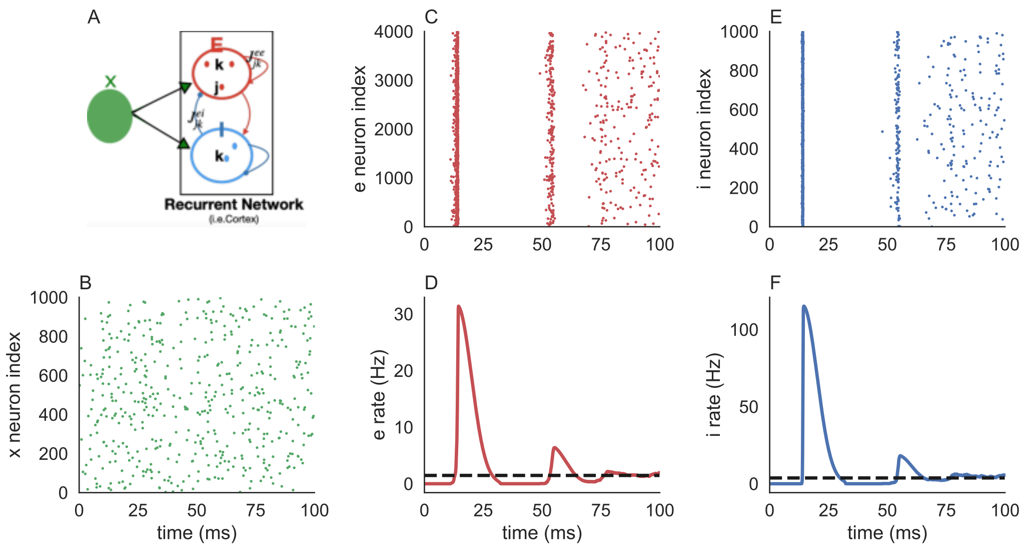

Figure 2.7. **EIF model in a recurrent network.** A: the network scheme with $N_x = 1000$ external neuron spikes using a Poisson process, and the recurrent network contains $N_e = 4000$ excitatory neurons and $N_i = 1000$ inhibitory ones. B, C, and E: roaster plots of 100 neurons from each group. D and F: trial-average firing rates of each post-synaptic neuron group and their mean-field rates in dashed lines.

2.3.2 Connectivity

To absorb the information about the size of each pre-synaptic neuron population and their connection strength, we now define a new N by N **connectivity matrix** (a.k.a. weight matrix, later a.k.a. mean-field synaptic weight, *or* normalized mean-field synaptic weight). Likewise, W also has a block structure where W_{jk}^{ab} is of the size $N_a \times N_b$ such that

$$W = \begin{bmatrix} W^{ee} & W^{ei} \\ W^{ie} & W^{ii} \end{bmatrix}$$

The value W_{jk}^{ab} represents the synaptic connection from the pre-synaptic neuron k in population $b = e, i$ to a post-synaptic neuron j in the population $a = e, i$ of the network. To connect with the synaptic weight matrix, J_{jk}^{ab} described above, the value of W_{jk}^{ab} is therefore $N_b j_{ab} p_{ab}$ since $W_{jk}^{ab} = N_b E[J_{jk}^{ab}]$.

A biologically realistic W_{jk}^{ab} is based on Dale's Law, so it should not have columns of the opposite sign, so *either* all "+", *or* all "-" inclusive with 0s and looks like the following,

$$W = \begin{bmatrix} + & + & \dots & - & - \\ + & + & \dots & - & - \\ \dots & & & & \\ + & + & \dots & - & - \end{bmatrix}$$

A spiking recurrent network is computationally expensive. Its memory and simulation run time grows like $\mathcal{O}(N^2)$, since for each neuron there are two loops–one is for the synaptic current update and the other one is for the membrane potential. To study their spike activities, it is more efficient to study the firing rate directly with the connectivity matrix instead of the rate records from the spiking model simulations.

2.3.3 Rate Model Approximations

To get the firing rate information of a network of neurons, besides modeling each neuron's membrane potentials and recording their spikes like in Eq. 2.18 systematically, it is often convenient to model their firing rate responses directly using the tool of "f-I" curve and connectivity matrix. In a mean-field input approximation along time, the average firing rate across the network population has an O.D.E. dynamical trajectory like

$$\tau \frac{d\mathbf{r}}{dt} = -\mathbf{r} + f(\bar{I}). \tag{2.19}$$

This is the standard **rate model** form, where τ is a timescale constant that represents the combined effect from the synaptic current and membrane potential dynamics in Eq. 2.18. Using the rate model in Eq. 2.19 to describe the EIF spiking model in Eq. 2.18, we have the firing rate dynamics for the recurrent network,

$$\tau \frac{d\mathbf{r}}{dt} = -\mathbf{r} + f(W\mathbf{r} + \mathbf{X}) \tag{2.20}$$

where τ is around $10 \sim 50$ms. where $\mathbf{r} = \begin{bmatrix} r_e \\ r_i \end{bmatrix}$, $W = \begin{bmatrix} W_{ee} & W_{ei} \\ W_{ie} & W_{ii} \end{bmatrix}$, and $\mathbf{X} = W_x r_x = \begin{bmatrix} W_{ex} r_x \\ W_{ix} r_x \end{bmatrix}$. Since W is block-wised as mentioned before, we have the corresponding mean-field synaptic connections, $W_{ab} = N_b j_{ap} p_{ab}$ where $a, b = e, i$.

To explain the dynamical parts of this rate model, we need to understand the stability concept through its **fixed point**. In other words, when the firing rates do not change over time, the fixed point solution is when there are no dynamics $\frac{d\mathbf{r}}{dt} = 0$. For example, the fixed point solution for Eq, 2.20 is

$$\mathbf{r} = f(\bar{I}) = f(W\mathbf{r} + \mathbf{X}). \tag{2.21}$$

In terms of computations, the rate model is better than the spiking model, and it also provides a decent amount of neuron spiking accuracy. Figure 2.7D and F show that the population-averaged but time-dependent firing dynamics generated from a spiking EIF model in Eq. 2.18 eventually become **stable** and reach their mean-field stationary rate approximations in dash lines.

One might wonder what if the fixed point of the firing rate model in the recurrent network is not stable. It is useful to study the Jacobian matrix, Jac, of the rate model in Eq. 2.21 as

$$Jac = \frac{1}{\tau}[-I + GW] \tag{2.22}$$

where $G = \text{diag}(f'(\bar{I}))$ is a diagonal matrix with entries $G_{jj} = f'(\bar{I}_j)$ for j identifies the neuron j in the recurrent network, and f is applied to each neuron mean-current input point-wise.

If all the eigenvalues of the Jacobian matrix, Jac, have a negative real part, then the fixed point solution is stable. If our $G = id$ in Eq. 2.22, this is also equivalent to have the real part eigenvalue of W less than 1. Figure 2.8A, B, and C show the two eigenvalues of the fixed points from Eq. 2.20 in the population level. Since we have two groups (one for excitatory and one for inhibitory) within the recurrent network, we have two eigenvalues. In general, we can also apply it to the neuron level. Figure 2.8D, E and F present the 100 eigenvalues from another RNN with 100 neurons, and we see that it contains some unstable fixed points as some points are outside of the stable bound (in red).

2.3.4 Balanced Network Theory

The computation time of an EIF spiking simulation in a recurrent network of the size $N = 5000$ ($N_e = 4000, N_i = 1000$) in Figure 2.7 takes more than 2 minutes in a regular computer. In reality, the cortical area contains much more neurons than the

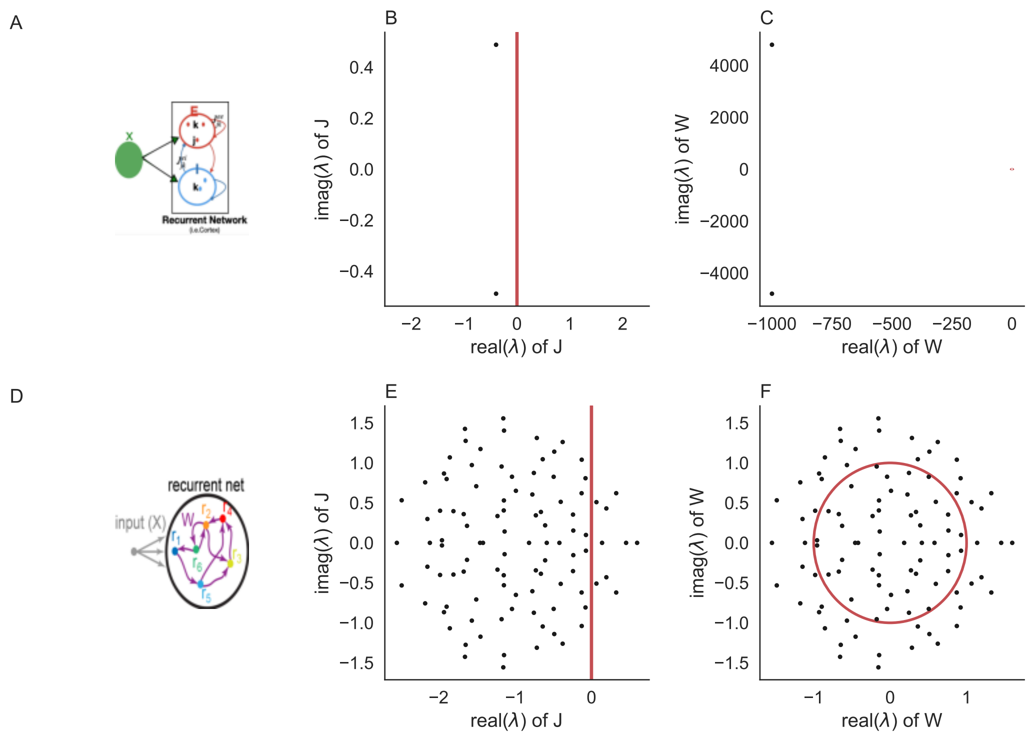

Figure 2.8. **Stability of the fixed point in RNNs.** A: BRNN in Section 2.3.1 under EIF model simulations and using a mean-field approximation of rate model in Eq. 2.19 with $\tau_e = 40ms$ and $\tau_i = 20ms$ to obtain population firing rates. B: the eigenvalue of the Jacobian matrix plot, $\mathrm{Re}\{\lambda(Jac)\} < 0$, so A has stable fixed points for each population. C: Same as in B, except for the weight matrix, W. A, B, and C are the population level. D: an RNN network with a size of $N = 100$ neurons directly under the rate model simulations. E, and F: same as in A, B, and C, except RNN contains some unstable fixed points for individual neurons in D, they are points on the right side of the zero bar in E and outside of the unit circle in F. D, E, and F are the neuron level, where W is generated from a random matrix.

5000 simulations (imagine the computation difficulties since the brain has 100 billion neurons). What happens to this network when the size increases? Of course, the mean-field synaptic weight W_{ab} should not go to infinity, and this implies that j_{ab} and p_{ab} need to carefully scale in order to keep the network in a "balance" mode. In fact, in Figure 2.7D and F, we have seen an example of "balance" when the excitatory and inhibitory firing rates reach to their stationary state, $\mathbf{r}_a \sim \mathcal{O}(1)$, where $a = e, i$. The **balanced network theory** is to study the excitatory and inhibitory "balanced" through some cancellations between their positive and negative input sources.

The mean-synaptic input of a recurrent network in a rate model (see Eq. 2.21)is $\bar{I} = W\mathbf{r} + \mathbf{X} \sim \mathcal{O}(NpJ)$. Within each population, the firing rate is assumed to be $\mathbf{r}_a \sim \mathcal{O}(1)$ where $a = e, i$. Since the "f-I" curve is the approximation of the firing rate through the network mean total current input ($r = f(\bar{I})$), it tells us to keep the mean-field total input still while increasing the network size N, so $\bar{I} \sim \mathcal{O}(1)$ as $N \to \infty$. This implies that we *either* have **sparsely coupled networks** such that the connectivity probability between one neuron population to another population must be small, like $p_{ab} \sim \mathcal{O}(\frac{1}{N})$, *or* the **weakly coupled networks** where the synaptic connection strength between neuron types are small (like $j_{ab} \sim \mathcal{O}(\frac{1}{N})$). This is equivalent to saying that the network size does not change the mean current input, but the spread of neurons.

In a weakly coupled network, the variance of each synaptic input is in an order of $\mathcal{O}(N_b p_{ab} j_{ab}^2 r_b)$. It would vanish as the network grows, since $Var(I_{ab}) \sim \mathcal{O}(\frac{1}{N})$). This indicates that the pre-synaptic neurons are uncorrelated and hence no spike train in a large network limit. However, studies have shown that this is inconsistent with real neuron recordings [82, 83]. Other scientists proposed that the synaptic strength should scaled like $j_{ab} \sim \mathcal{O}(\frac{1}{\sqrt{N}})$ with $p_{ab} \sim \mathcal{O}(1)$. This scaling is to have a non-vanishing variance and makes it biologically realistic, so it is called **strongly coupled networks**. If we work with this scaling scheme, it implies that the mean total inputs would be in the order of $\sqrt{N}$, since each means synaptic current input has the expression $\bar{I}_{ab} = W_{ab}r_b = N_b p_{ab} j_{ab} r_b \sim \mathcal{O}(\sqrt{N})$. However, the original assumption is that the total mean current is moderate, $\bar{I} \sim \mathcal{O}(1)$. This implies that there must be some cancellations to compensate for that.

We first defined a **normalized mean-field synaptic weights** as $\tilde{W}_{ab} = \frac{N_b p_{ab} j_{ab}}{\sqrt{N}}$, where $a = e, i$ and $b = e, i, x$, then each mean current input is $\bar{I}_{ab} = \sqrt{N}\tilde{W}_{ab}r_b \sim \mathcal{O}(\sqrt{N})$ since $\tilde{W}_{ab}r_b \sim \mathcal{O}(1)$. In order to achieve a **moderate** total mean current input, $\bar{I} \sim \mathcal{O}(1)$, cancellation must exist when combining these terms. Specifically, this implies that the network can receive **strong** synaptic input from each input source.

In another scope, the notion of "balance" can be viewed from the cancellation of the "local" recurrent network term, $\tilde{W}\mathbf{r}$ and the scaled external input term, $\tilde{\mathbf{X}}$ such that

$$\tilde{W}\mathbf{r} + \tilde{\mathbf{X}} \sim \mathcal{O}(\frac{1}{\sqrt{N}}) \tag{2.23}$$

In a large network limit, $N \to \infty$, Eq. 2.23 becomes

$$\lim_{N\to\infty} \tilde{W}\mathbf{r} + \tilde{\mathbf{X}} = 0 \tag{2.24}$$

We can rearrange terms and solve for the firing rate solution under the existence of $\tilde{W}^{-1}$,

$$\lim_{N\to\infty} \mathbf{r} = -\tilde{W}^{-1}\tilde{\mathbf{X}}. \tag{2.25}$$

Since the network must be finite, $N < \infty$, so we can write the firing rates as an approximate, $\mathbf{r} \approx -\tilde{W}^{-1}\tilde{\mathbf{X}}$. Note that in general, the external stimulus, $\tilde{\mathbf{X}}$ is positive, so it is equivalent to having another excitatory neuron group. Biologically, the firing rate must be positive, $\mathbf{r} > 0$, this indicates that $\tilde{W}\mathbf{r} < 0$, so the recurrent part of this network is inhibitory.

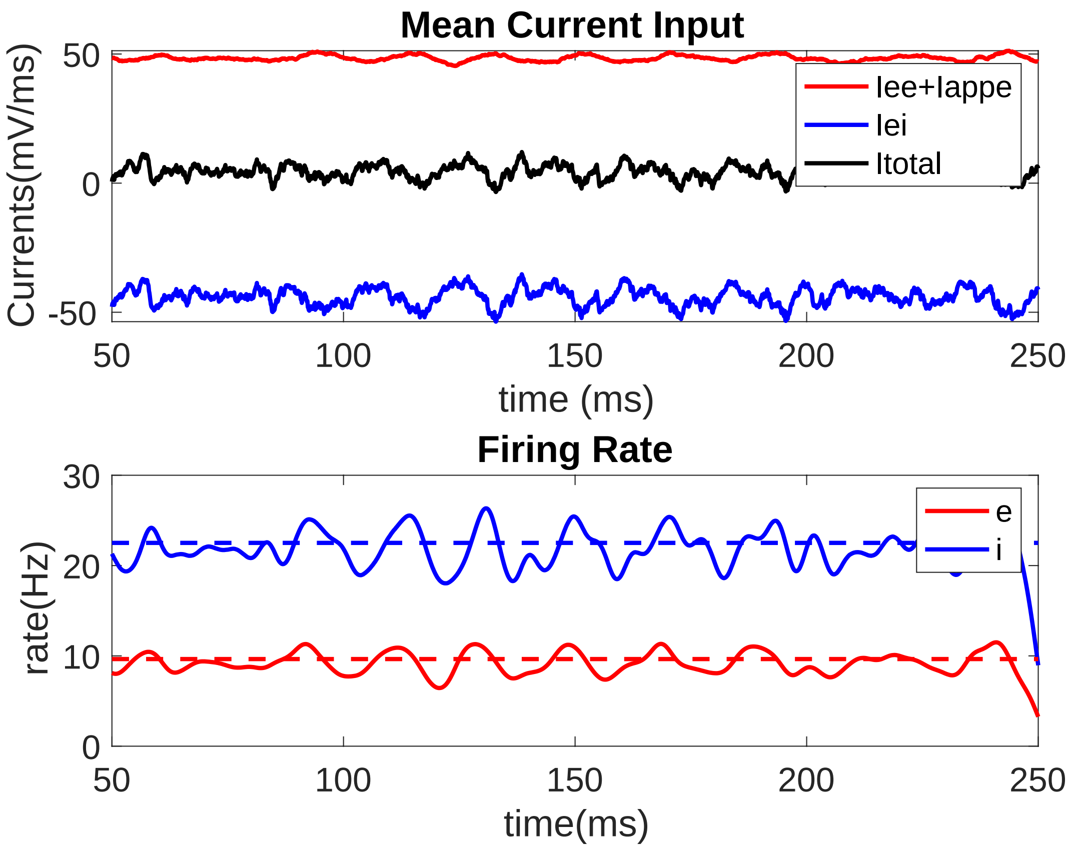

Figure 2.9. **E-I Balanced in a recurrent network simulated from an EIF spiking model.** Top: mean current input into excitatory (in red, positive) and inhibitory (in blue, negative) population within the recurrent network. The mean total input (in black) is the summation of all the inputs, and it is bouncing around 0. Bottom: trial-average firing rates (in solid colored lines) for each neuron population and their stationary mean-field rates (in dashed colored lines).

Despite the “local”-external balance for the recurrent network, one can also separate the mean total input current as excitatory ($\bar{I}_{ae}+\bar{I}_{ax}$) and inhibitory ($\bar{I}_{ai}$) components. While each mean current input source is **strong**, $\bar{I}_{ab} \sim \mathcal{O}(\sqrt{N})$ where $a = e, i$ and $b = e, i, x$, the average current flows to the excitatory and inhibitory population within the recurrent network is still **moderate**, $\bar{I}_a = \bar{I}_{ae} + \bar{I}_{ai} + \bar{I}_{ax} \sim \mathcal{O}(1)$. We must have some cancellations to achieve the **excitatory-inhibitory balance** when adding these current sources. Figure2.9 demonstrates that even though the individual current source is strong, the total mean current input remains at zero. Notice that the balanced network theory and the “f-I” curve both model the mean-field approximations of the firing rates, but these two do not rely on each other.

2.4 Modeling Through Synaptic Plasticity and Learning

So far, we have shown models for a single neuron and their extension to a network of neurons. We have seen spiking models like EIF that can capture the membrane potential for a single neuron as well as multiple neurons in the mean-field approximation level. In addition, we were able to use the “f-I” curve and rate model dynamics to describe the firing rate responses in a relationship with mean total current inputs in recurrent networks. Furthermore, the stability analysis of the fixed point solutions from firing rate dynamics gives us another scope to understand the importance of neuron synaptic strength and the firing rate time scale. Notice that these models are for the purpose of *either* portraying a phenomenon of cortical neurons, like the spiking behavior, *or* describing a relationship between two quantities, like firing rate responses and mean current inputs. On a bigger scale, what if we want the brain actually “do” something? For example, can it identify an object or learn something new? In this section, we focus on building BRNN models for the purpose of “solving” some tasks.

2.4.1 Synaptic Plasticity

Many things change and evolve with time! In neural activities, **synaptic plasticity** happens when the strength of neuron synapses changes (ΔJ). The strengthening of synaptic connectivity is called **facilitation** while weakening is called **depression** over time. These changes can happen instantaneously (a.k.a. short-term plasticity) or over a lengthy period of time (a.k.a long-term plasticity). In particular, long-term plasticity is associated with the learning and memory mechanism in the brain.

Among all the types of plasticity rules, **hebbian plasticity** (named after Donald Hebb) is the most widely studied and well-known type of long-lasting activity changes in synaptic strength. The main idea of the Hebbian principle states that the magnitude of an increase in synaptic strength is proportional to the firing rates of pre-and post-synaptic neuron pairs [38]. This is often described as "neurons that fire together, wire together." Specifically, if there is an increase in the firing rate of pre-synaptic neuron k and resulting the post-synaptic neurons j spiking more often as well, then their synaptic connectivity J_{jk} will get stronger. Hence, to translate the Hebbian plasticity rule directly, we have

$$\frac{dJ_{jk}}{dt} = c\mathbf{r}_j\mathbf{r}_k \tag{2.26}$$

where c is the scale constant. However, Eq. 2.26 indicates that synaptic strength dynamics can grow infinitely, and this is certainly not biologically realistic because it leads to instability. In many contexts, the inhibition can stabilize recurrent networks, so we introduce a model with homeostatic **inhibitory synaptic plasticity (ISP)** that stabilizes our BRNNs. This model focuses on the synaptic strength dynamics of a single excitatory neuron e. It receives synaptic input from another inhibitory neuron i with the synaptic strength $J_{ei} < 0$ (since this synapse strength source is coming from inhibitory) such that

$$\begin{aligned}\frac{dJ_{ei}}{dt} &= -\eta\left[(y_e(t) - 2r_o) * S_i(t) - S_e(t) * y_i(t)\right] \\ \tau_y \frac{dy_a}{dt} &= -y_a(t) + S_a(t)\end{aligned} \tag{2.27}$$

where η is the **learning rate** that controls how quickly the synaptic strength J_{ei} changes, and $r_0 > 0$ is the **target rate** for neuron e. We define the **synaptic trace** as $y_a(t)$, with $a = e, i$ representing a single neuron, to measure the running estimates of the neuron firing rates. τ_y is the timescale constant for the synaptic trace dynamics. $S_a(t)$ is a spike train with two values (0 or 1), and it is modeled from a Poisson process with the mean-field firing rate r_a, J_{ei} gets updated only after a spike *either* from neuron e *or* neuron i. For example, Figure 2.10 depicts the changes in synaptic strength between a pair of pre-and post-synaptic neuron spikes. If neuron i spikes at time t_{pre}, so $S_i(t_{pre}) = 1$, then J_{ei} is reduced by $2\eta r_0$ amount because there is no spike for the post-synaptic neuron e yet at the time of t_{pre}, so $y_e(t_{pre}) = 0$, hence $S_e(t_{pre}) = 0$. Similarly, if neuron e spikes at time $t_{post} > t_{pre}$, then J_{ei} is changing by $-\eta y_i(t_{post})$ quantity (since $S_e(t_{post}) = 1$). Notice that $J_{ei} < 0$, so the "-" sign is pushing the J_{ei} dynamics more negative (inhibition), hence the synaptic strength gets stronger.

Figure 2.10C shows that the ISP rule is time-dependent. In general, plasticity rules that rely on the pre-and post-synaptic neuron pair spike time are called **spike time-dependent plasticity (STDP)** rules. To understand the effect of time evolution on synaptic strength, let us focus on the moment of $t_{post} > t_{pre}$ (since t_{pre} already happened). Notice that the second O.D.E. in Eq. 2.27 has an analytic solution as an exponential decay after the pre-synaptic neuron i spikes, so the trace of inhibitory neuron has the expression, $y_i(t_{post}) = \frac{1}{\tau_y} e^{-\frac{t_{post} - t_{pre}}{\tau_y}}$. According to the ISP rule, when neuron i spikes, it contributes to the change of synapse for the post-synaptic neuron e by $\Delta J_{ei} = 2\eta r_0$ amount. Since the pre-synaptic neuron i spike also leads to the

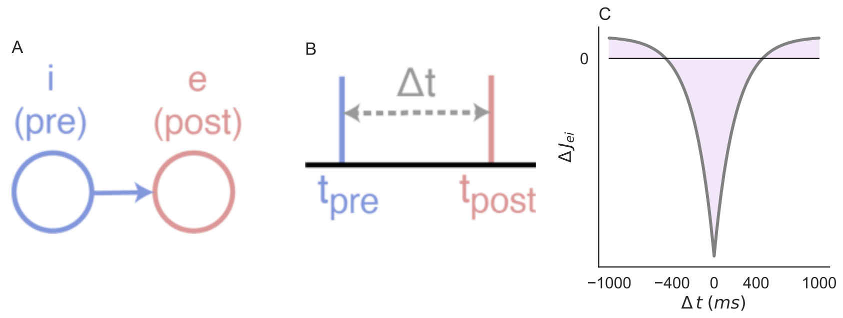

Figure 2.10. **ISP from an inhibitory neuron to an excitatory neuron** A: diagram of a synaptic neuron pair from inhibition to excitation. B: the spike time delay, Δt, between two spikes. C: the change of synaptic strength, ΔJ_{ei}, as a function of Δt.

post-synaptic neuron e spikes, so neuron e itself also contributes $\Delta J_{ei} = -\eta y_i(t_{post})$ amounts. Hence, the total update in synaptic strength for neuron e looks like

$$\Delta J_{ei} = 2\eta r_0 - \frac{\eta}{\tau_y} e^{-\frac{t_{post}-t_{pre}}{\tau_y}} \tag{2.28}$$

Now the next stage is when neuron e becomes a new spiking pre-synaptic neuron, then following the same logic, we have $\Delta J_{ei} = 2\eta r_0 - \frac{\eta}{\tau_y} e^{-\frac{t_{pre}-t_{post}}{\tau_y}}$. Combine above equations all together, we have

$$\Delta J_{ei} = 2\eta r_0 - \frac{\eta}{\tau_y} e^{-\frac{|\Delta t|}{\tau_y}} \tag{2.29}$$

where $\Delta t = t_{post} - t_{pre}$ is the **time delay** between a pre-synaptic neuron spike and a post-synaptic neuron spike. In a more realistic setting, we can transfer the STDP rule for a single neuron to a recurrent network under the EIF spiking model that we discussed in Section 2.3.1, and Eq. 2.18 can be developed as

$$\begin{aligned}
\tau_m \frac{d\mathbf{V}^a}{dt} &= -(\mathbf{V}^a - E_L) + De^{\frac{\mathbf{V}^a - V_t}{D}} + \mathbf{I}^{ae}(t) + \mathbf{I}^{ai}(t) + \mathbf{I}^{ax}(t) \\
\tau_b \frac{d\mathbf{I}^{ab}}{dt} &= -\mathbf{I}^{ab}(t) + \mathbf{J}^{ab}\mathbf{S}^b(t) \\
\text{if } \mathbf{V}_j(t) &> \mathbf{V}_{th}, \text{record a spike at time t; then reset } \mathbf{V}_j(t) \to \mathbf{V}_{re}. \\
\tau_y \frac{d\mathbf{y}^a}{dt} &= -\mathbf{y}^a(t) + \mathbf{S}^a(t) \\
\frac{d\mathbf{J}^{ei}}{dt} &= -\eta \left[(\mathbf{y}^e(t) - 2r_o) \left[\mathbf{S}^i(t)\right]^T - \mathbf{S}_e(t) \left[\mathbf{y}_i(t)\right]^T \right] \circ \Omega^{ei}
\end{aligned} \tag{2.30}$$

where Ω^{ei} is an indicator matrix with entries valued at *either* zero *or* one as the following,

$$\Omega^{ei}_{jk} = \begin{cases} 1 & \text{when there is an initial connection } J^{ei}_{jk}(0) \neq 0 \\ 0 & \text{otherwise} \end{cases}.$$

Here, $\circ$ means matrix element-wise multiplication and indicates that the update only applies to the neurons that actually have connectivity strengths with other neurons within this recurrent network.

Next, we want to use the knowledge from mean-field theory to explain and approximate the firing rates generated from the EIF spiking model in Eq. 2.30 in the dynamical rate model from Eq. 2.20by including the weight updates as

$$\begin{aligned}
\tau \frac{d\mathbf{r}}{dt} &= -\mathbf{r} + f(W\mathbf{r} + \mathbf{X}) \\
\frac{dw_{ei}}{dt} &= -\eta_r (r_e - r_0) r_i.
\end{aligned} \tag{2.31}$$

Again, the connectivity entry w_{ei} absorbs the synaptic strength J^{ei} since in section 2.3.2. In a homogeneous setting, the connectivity matrix entry for recurrent networks is $w^{ei}_{jk} = N_i E[J^{ei}_{jk}] = N_i J_{ei} p_{ei}$. Hence, the corresponding learning rate would be $\eta_r \propto N_i p_{ei} \eta$. Moreover, the expectation of trace is the same as the spiking target

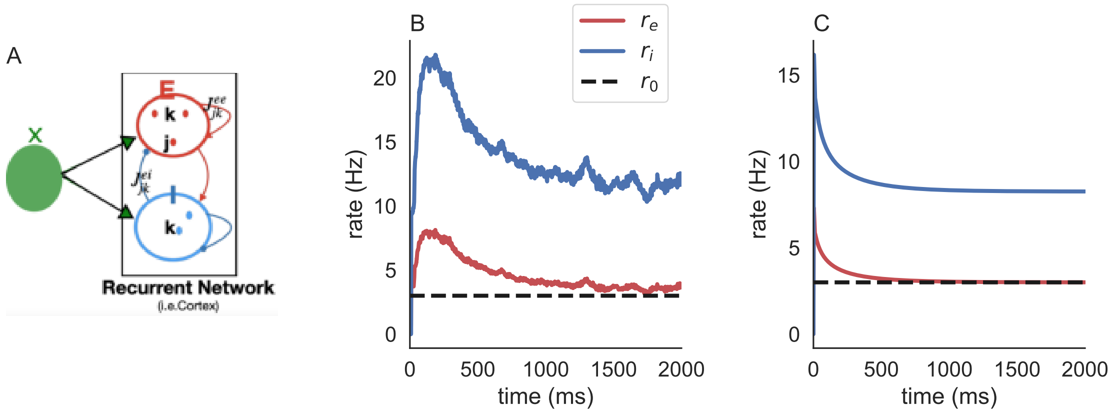

Figure 2.11. **ISP under an EIF spiking model in a recurrent network** A: diagram of a recurrent spiking network using the same parameter setting as in Figure 2.7. B: firing rates of dynamics from the EIF spiking model evolves through ISP: the inhibitory firing rate (in blue) pushes the excitatory firing rates (in red) towards its target rate (in black, r_0. C: a mean-field approximation rate model of B.

rate, r_a ($= E[S^a(t)] = E[y^a(t)]$) where $a = e, i$. If we solve the weight dynamics O.D.E. in Eq. 2.31 for the fixed point solutions, then we must *either* have $r_e = r_0$ *or* $r_i = 0$ (note that this is not possible since the inhibitory neurons initiate the spikes). Therefore, under stability conditions, r_e always reaches its target rate at r_0 here through ISP.

2.4.2 Learning

Learning can be a response to the change in synaptic strength. Through ISP in Section 2.4.1, our recurrent network "learns" to approach its target rate in the EIF spiking model as well as the mean-field approximated rate model. Although learning to reach a constant value is rather a simple task, the complex brain can obviously solve more difficult tasks in a real-world setting.

In the following chapters, we develop and present more complex tasks that our models in BRNNs can “do”. Some of our models are more biologically plausible such as spiking models with ISP, while others focus on neural activities which can be replaced by using the mean-field approximations of the rate models. Moreover, some models that we designed can identify objects, while others can detect errors. Furthermore, to link with ARNNs in machine learning, we also compare the learning tasks’ performance through algorithms that are inspired by ARNNs such as the gradient-based method with our newly developed learning rules.

Let us go to the Odyssey of brain-learning chapters!

CHAPTER 3

UNIVERSAL PROPERTIES OF STRONGLY CONNECTED NETWORKS

This chapter is partially adapted from [10].

One fundamental difference between BRNNs and most computational models is that cortical networks in the brain are multi-layered and recurrent (like what we see in section 2 Figure 1.2, right or Figure 1.3), while most computational models are either multi-layered or recurrent (Figure 1.2, left and middle). The cerebral cortex uses multiple layers of locally recurrent layers to process both dynamic and static stimuli. These observations raise the following question: How does the recurrent, intralaminar connectivity of cortical circuits shape their representation of stimuli? This question has been approached using a wide variety of different computational models such as rate models Eq. 2.19 in section 2. A mean-field approach with a proper scaling technique can produce asynchronous-irregular activity and support excitatory-inhibitory balance theory, which has also been widely observed in cortical recordings.

In Dr. Rosenbaum's neural dynamics and computing lab with a former Ph.D. student Dr. Baker, we showed that partial breaks in excitatory-inhibitory balance produce a characteristic non-linear relationship between the total current input and fixed points of the firing rate dynamics. Specifically, I formalized the features of cortical networks into mathematical assumptions and proved several theorems. This work presents the nonlinear stimulus representations and nonlinear computations, which are unavoidable in networks driven by multiple stimuli. These theorems are also consistent with real cortical recordings from experiments, and have a direct mathematical relationship to ANNs in machine learning.

3.1 Introduction

An approximate **balanc**e between excitatory and inhibitory synaptic currents is widely observed in cortical recordings [1, 11, 28, 36, 63, 95]. How do this balance shape neural computations and stimulus representations? This problem is often studied using computational models of neuronal networks in a dynamically balanced state. Despite the complexity of spike timing dynamics in the models of a dynamically stable balanced state, their population level firing rates [29, 52, 72, 82, 83] and correlations [25, 26, 39, 71, 73, 93] in response to given stimulus can be derived using a simple mean-field theory from Section 2.3.3. In this section, We address these problems with balanced network theory by developing a theory of **semi-balanced** networks that quantify network responses when the classical balanced network state is broken.

The classical theory of balanced networks has several shortcomings. First, it predicts a linear relationship between stimuli and neural population responses, in contrast to the nonlinear computations that must be performed by cortical circuits. Secondly, parameters in balanced network models must be chosen so that the firing rates predicted by balanced network theory are non-negative. In the widely studied case of one excitatory and one inhibitory population, parameters for network connectivity and external input must satisfy only two inequalities to achieve positively predicted rates[39, 83]. However, strictly positive predicted rates can be more difficult to achieve in networks with several populations such as multiple neuron subtypes, neural assemblies, or tuning preferences [52, 67]. This difficulty occurs because the proportion of parameter space for which predicted rates are non-negative becomes exponentially small with an increasing number of populations. Moreover, a given network architecture might produce a balanced state for some stimuli, but not others. Indeed, we show that for any network architecture satisfying Dale's law, there are infinitely many excitatory stimuli for which balanced network theory predicts negative rates, implying that any network structure admits stimuli that break the classical balanced state.

In the semi-balanced state, balance is only enforced in one direction: neurons can receive excess inhibition, but not excess excitation. Neurons receiving excess inhibition are silenced and the remaining neurons form a balanced sub-network. We show that semi-balanced networks implement nonlinear stimulus representations and computations. Specifically, we establish a mathematical relationship between semi-balanced networks and ANNs used for machine learning [34], as well as threshold-linear networks studied for their rich dynamics [23, 24, 35, 94]. We show that semi-balance, but not balance, is naturally realized at a neuron-by-neuron level in networks with homeostatic inhibitory plasticity [40, 84]. In this setting, semi-balanced networks have extended the traditional notion of excitatory-inhibitory balance in which BRNNs implement high-dimensional, richer, and nonlinear stimulus representations.

In summary, in contrast to the classical balanced state, the semi-balanced state is realized naturally in networks with time-varying stimuli, produces nonlinear stimulus representations, and has a direct correspondence to ANNs used in machine learning. The theory of semi-balanced networks, therefore, has extensive implications for understanding stimulus representations and computations in cortical circuits.

3.2 Spiking Network Model Descriptions

We consider a recurrent network of $N = 3 \times 10^4$ (80% excitatory and 20% inhibitory neurons altogether) randomly connected adaptive EIF neuron models. We chose the **adaptive EIF** neuron model because it is simple and efficient to simulate while also being biologically realistic [16, 44]. This current-based model used in all figures, and the membrane potential of neuron $j = 1, \ldots, N_a$ in population a obeyed

$$\begin{aligned}\tau_m \frac{dV_j^a}{dt} &= -(V_j^a - E_L) + D_T e^{\frac{V_j^a - V_T}{D_T}} - w + I_j^a(t) \\ \tau_w \frac{dw_j^a}{dt} &= -w_j^a\end{aligned} \tag{3.1}$$

with the added condition that each time $V_j^a(t)$ crossed $V_{th} = 0\text{mV}$, a spike was recorded, it was reset to $V_{re} = -72\text{mV}$, and w_j^a was incremented by $B = 0.75\text{mV}$. A hard lower bound was imposed at $V_{lb} = -85\text{mV}$. Other neuron parameters were $\tau_m = 15\text{ms}$, $E_L = -72\text{mV}$, $D_T = 1\text{mV}$, $V_T = -55\text{mV}$, and $\tau_w = 200\text{ms}$. The input was given by

$$I_j^a(t) = \sum_b \sum_k J_{jk}^{ab} \sum_n \alpha_b(t - t_{k,n}^b)$$

where $t_{k,n}^b$ is the nth spike of neuron k in population b and $\alpha_b(t) = \frac{e^{-\frac{t}{\tau_b}}}{\tau_b} H(t)$ is an exponential postsynaptic current with $H(t)$ the Heaviside step function. Synaptic time constants, τ_b, were 8/4/10ms for excitatory/inhibitory/external neurons. Synaptic weights were generated randomly and independently by

$$J_{jk}^{ab} = \begin{cases} \frac{j_{ab}}{\sqrt{N}} & \text{with probability } p_{ab} \\ 0 & \text{otherwise} \end{cases}$$

The network receives feedforward synaptic input from two external populations of Poisson processes, modeling external synaptic input in Figure 3.1a. The firing rates, $\mathbf{r}_x = [r_{x1}, r_{x2}]^T$, of the external populations form a two-dimensional stimulus space (Figure 3.1b $\mathbf{v}^T$ denotes the transpose of $\mathbf{v}$).

3.2.1 Simulations of An Adaptive EIF Model

In Figure 3.1 and 3.2, external input rates were $\mathbf{r}_x = [15, 15]^T\text{Hz}$ for the first 500ms and $\mathbf{r}_x = [15, 30]^T\text{Hz}$ for the next 500ms.

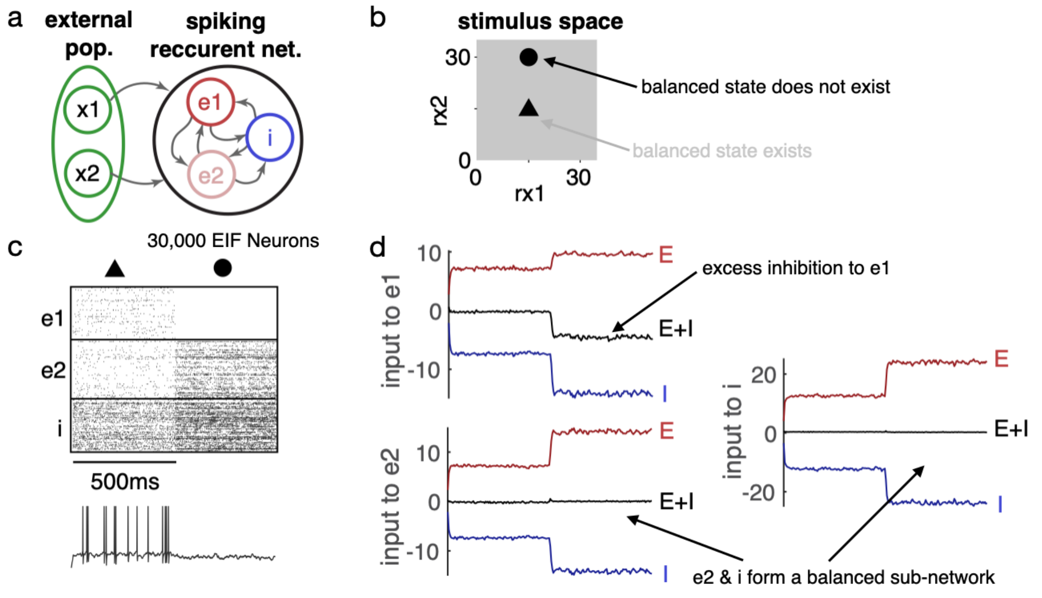

Figure 3.1. **Balanced and Semi-balanced States** a: Network Diagram. A recurrent spiking network of $N = 3 \times 10^4$ model neurons is composed of two excitatory populations ($e1$ and $e2$) and one inhibitory population (i) that receive input from two external spike train populations ($x1$ and $x2$). b: The two-dimensional space of external population firing rates represents a stimulus space. The filled triangle and circle show the two stimulus values used in d the first half of 500ms and the second half of 500ms correspondingly. c: Raster plots of 200 randomly selected spike trains from each population for two stimuli, and below is the membrane potential of one neuron from population $e1$. d: Mean input current to population $e1$, $e2$, and i from all excitatory sources ($e1, e2, x1$, and $x2$; red), from the inhibitory population (i; blue), and from all sources (black) showing approximate excitatory-inhibitory balance across stimuli in the first half of 500ms. With excess inhibition in the second half of 500ms, the semi-balance state shows that $e2$ and i population form a balanced sub-network.

In Figure 3.1 and 3.2, postsynaptic populations were $a = e1, e2, i$ and pre-synaptic populations were $b = e1, e2, i, x1, x2$ with $N_{e1} = N_{e2} = 1.2 \times 10^4$, $N_i = 6000$, and $N_{x1} = N_{x2} = 3000$ so that $N = N_{e1} + N_{e2} + N_i = 3 \times 10^4$. Neurons in external populations, $x1$ and $x2$, were not modeled directly, but spike times were generated as independent Poisson processes with firing rates r_{x1} and r_{x2}. Connection strength coefficients were $j_{ejek} = 0.375$, $j_{eji} = -2.25$, $j_{iek} = 1.70$, $j_{ii} = -0.375$, $j_{ejxk=2.70}$, $j_{ixk} = 2.025$mV/Hz for $j, k = 1, 2$. Note that these were scaled by $\sqrt{N}$ to get the actual synaptic weights as defined above. Note that some balanced network studies scale weights by $\sqrt{K}$ instead of $\sqrt{N}$. Since we keep connection probabilities fixed, $K \sim N$, so scaling by $\sqrt{N}$ is equivalent to scaling by $\sqrt{K}$. This choice of synaptic weights produced postsynaptic potential amplitudes between 0.07mV and 0.8mV. Connection probabilities in Figure 3.1 and 3.2 were $p_{e1e1} = p_{e2e2} = 0.15$, $p_{e1e2} = p_{e2e1} = 0.05$, $p_{e1x1} = 0.08$, $p_{ix1} = p_{ix2} = 0.12$, and $p_{ab} = 0.1$ for all other connection probabilities.

3.2.2 Simulations with Inhibitory Plasticity

For Figure 3.3, the model was the same as above except there was just one excitatory, one inhibitory, and one external population with $N_e = 0.8N$ and $N_i = N_x = 0.2N$ where $N = 3{\times}10^4$. Stimulus coefficients in Fig 3B were set to $\sigma_1 = \sigma_2 = 22.5$mV (about 1.4 times the rheobase) for the first 80s and randomly selected from a uniform distribution on $[-30, 30]$mV for the last 40s. Connection probabilities between all populations in Figure 3.3 were $p_{ab} = 0.1$. Initial synaptic weights were given by $j_{ee} = 37.5$, $j_{ei} = -225$, $j_{ie} = 168.75$, $j_{ii} == 375$, $j_{ex} = 2700$, and $j_{ix} = 2025$mV/Hz as above. Only inhibitory weights onto excitatory neurons (j_{ei}) changed, all others were plastic.

The **inhibitory plasticity** rule was taken directly from previous work [84]. The variables, $x^a_j(t)$ represent filtered spiking activity and are defined by $\tau_x \frac{dx^a_j}{dt} = -x^a_j$

with the added condition that $x_j^a(t)$ was incremented by one each time neuron j in population $a = e, i$ spiked. After each spike in excitatory neuron j, inhibitory synaptic connections onto that neuron were updated by $\Delta J_{jk}^{ei} = -\eta x_k^i(t)$ for all non-zero J_{jk}^{ei}. After each spike in inhibitory neuron, k, its outgoing synaptic connections were updated by $\Delta J_{jk}^{ei} = -\eta(x_k^i(t) - \alpha)$. We used $\tau_x = 200$ms and $\alpha = 2$ to get a "target rate" of $r_e^t = \frac{\alpha}{2\tau} = 5$Hz.

3.3 Linear Representations in Balanced Networks

To review balanced network theory from Section 2.3.4 and its limitations, simulations in Figure 3.1C of this model showed asynchronous-irregular spiking activity and excitatory-inhibitory balance. How does connectivity between the populations determine the mapping from the stimulus, r_x, to firing rates, $\mathbf{r}_x = [r_{e1}, r_{e2}, r_i]^T$ in the recurrent network? Firing rate dynamics in Eq. 2.21 can be approximated using models of the form

$$\tau\dot{\mathbf{r}} = -\mathbf{r} + f(\overline{JK}[W\mathbf{r} + \mathbf{X}]) \tag{3.2}$$

where $\dot{\mathbf{r}}$ denotes the time derivative, f is a non-decreasing f-I curve, and W is the effective recurrent connectivity matrix. External input is quantified by $\mathbf{X} = W_x\mathbf{r}_x$. Components of W is given by $w_{ab} = \frac{J_{ab}K_{ab}}{\overline{JK}}$ where K_{ab} is the mean number of connections from population b to a and J_{ab} is the average connection strength. The coefficient, $\overline{JK} = \text{average}(|J_{ab}|K_{ab})$, quantifies coupling strength in the network. Since $\overline{JK}$ is multiplied in the equation for $\dot{\mathbf{r}}$ and divided in the equation for w_{ab}, it does not affect dynamics but serves as a rotational tool in the calculation below, which require $\overline{JK} \sim J_{ab}K_{ab}$ so that $w_{ab} \sim \mathcal{O}(1)$ even when $J_{ab}K_{ab}$ is large.

The key idea underlying balanced network theory is that $\overline{JK}$ is typically large in cortical circuits because neurons receive thousands of synaptic inputs and each postsynaptic potential is moderate in magnitude. Total synaptic input

$$\mathbf{I} = \overline{JK}[W\mathbf{r} + \mathbf{X}] \tag{3.3}$$

can only remind $\mathcal{O}(1)$ if there is a cancellation between excitation and inhibition. In particular, to have $\mathbf{I} \sim \mathcal{O}(1)$, we must have $W\mathbf{r} + \mathbf{X} \sim \mathcal{O}(\frac{1}{\overline{JK}})$, so in the limit of large $\overline{JK}$, firing rates satisfy

$$\mathbf{r} = -W^{-1}\mathbf{X} \tag{3.4}$$

In classical balanced network theory, one considers the $N \to \infty$ limit while taking $J_{ab} \sim 1/\sqrt{N}$ and $K_{ab} \sim N$ so that $\overline{JK} \to \infty$ and Eq. 3.4 is exact in the limit [83]. Experimental evidence for this scaling has been found in cortical cultures [11]. Note that while Eq. 3.2 is a heuristic approximation to spiking networks, the conclusion that Eq. 3.4 must be satisfied to keep $\mathbf{I} \sim \mathcal{O}(1)$ as $\overline{JK} \to \infty$ does not depend on the approximation in Eq. 3.2, but is implied by Eq. 3.3 alone and is therefore mathematically valid for spiking networks [83] for which firing rates can depend on the variance, and higher order moments of neurons' synaptic input. Even though it is derived as a limit, Eq. 3.4 provides a simple approximation to firing rates in networks with finite $\overline{JK}$. Indeed, it accurately predicted firing rates in our spiking network simulations for which $\overline{JK} = 5.9$ mV/Hz.

While the simplicity of Eq. 3.4 is appealing, it **linearly** reveals a critical limitation of balanced networks as models of cortical circuits: Because $\mathbf{r}$ depends linearly on $\mathbf{X}$ and $\mathbf{r}_x$, balanced networks can only implement linear representations of stimuli and linear computations [2, 29, 83].

To demonstrate this linearity in our spiking network, we sampled a lattice of points in the two dimensional space of $\mathbf{r}_x = [r_{x1}, r_{x2}]^T$ values and plotted the resulting neural **manifold** traced out in three dimensions by $\mathbf{r} = [r_{e1}, r_{e2}, r_i]^T$. The resulting manifold is approximately linear, *i.e.,* a **plane** in Figure 3.2b because $\mathbf{r}$ depends linearly on

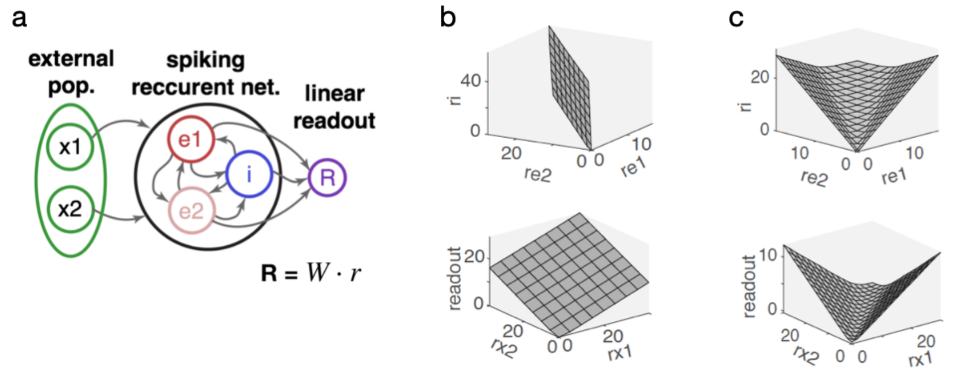

Figure 3.2. **Firing Rates Representation in Balanced and Semi-balanced Networks** a: Network Diagram, same as in Figure 3.1 except included a linear readout R output. b: top is the neural manifold traced out by firing rates in each population in the recurrent network as external firing rates are varied across a square in stimulus space ($0 \leq r_x 1, r_x 2 \leq 30$), and the bottom is the readout as a function of $r_x 1$ and $r_x 2$ from the same simulation. c: Same as c, except this is for the semi-balanced state. All firing rates are in Hz.

$\mathbf{X}$, and therefore on $\mathbf{r}_x$, in Eq. 3.4. More generally, the neural manifold is the n_x-dimensional hyperplane in n-dimensional space where n and n_x are the numbers of populations in the recurrent and external populations respectively. In addition, any linear readout $R = \mathbf{W} \cdot \mathbf{r}$ is a linear function of $\mathbf{r}_x$ and therefore also planar in Figure 3.2b.

How do cortical circuits, which exhibit excitatory-inhibitory balance, implement nonlinear stimulus representations and computations? Below, we describe a parsimonious generalization of balanced network theory that allows for nonlinear stimulus representations by allowing excess inhibition without excess excitation.

3.4 Nonlinear Representations in Semi-balanced Networks

Note that Eq. 3.4 is only valid if all elements of $\mathbf{r}$ it predicts are non-negative. Early work considered a single excitatory and single inhibitory population, in which case positivity of $\mathbf{r}$ is assured by simple inequalities satisfied in a large proportion of parameter space [72, 83]. Similarly, in the simulations described above, we constructed W and W_x so all components of $\mathbf{r}$ were positive for all values of $r_{x1}, r_{x_2} > 0$.

3.4.1 Conditions Break the Classical Balanced State

In networks with a large number of populations, conditions to assure $\mathbf{r} > 0$ become more complicated and the proportion of parameter space satisfying $\mathbf{r} > 0$ becomes exponentially small. In addition, we proved that connectivity structures, W, obeying Dale's law necessarily admit some positive external inputs, $\mathbf{X} > 0$, for which Eq. 3.4 predicts negative rates (see **Theorem 1** for proof details). Hence, the classical notion of excitatory-inhibitory balance cannot be assured by conditions imposed on the recurrent connectivity structure, W, alone, but conditions on stimuli, $\mathbf{X}$ and $\mathbf{r}_x$ are also needed.

Theorem 1 *Suppose W is a real, non-singular $n \times n$ matrix. for which each column is either non-negative or non-positive (Dale's law), each column has at least one non-zero element, and there is at least one positive entry in the matrix. Then there exists an $n \times 1$ vector, $\boldsymbol{X}$, with strictly positive entries ($\boldsymbol{X}_j > 0$ for all j) for which the $n \times 1$ vector defined by $\boldsymbol{r} = -W^{-1}\boldsymbol{X}$ has at least one negative entry ($\boldsymbol{r}_j < 0$ for some j).*

Proof. Without loss of generality, we can rearrange columns to write W with the non-negative columns first and the non-positive ones next,

$$W = \begin{bmatrix} + & + & \dots & - & - \\ + & + & \dots & - & - \\ \dots & & & & \\ + & + & \dots & - & - \end{bmatrix}$$

where each $+$ is an element that is ≥ 0 and each $-$ is ≤ 0. Now define an $n \times 1$ column vector

$$v = \begin{bmatrix} - \\ - \\ \dots \\ + \\ + \end{bmatrix}$$

where each $-$ is a negative number, each $+$ is a positive number, there are the same number $-$ in v as there $+$ columns in W, and the same number of $+$ in v as $-$ entries in W. Finally, define

$$\begin{aligned} \mathbf{X} &= -Wv \\ &= -\begin{bmatrix} + & + & \dots & - & - \\ + & + & \dots & - & - \\ \dots & & & & \\ + & + & \dots & - & - \end{bmatrix} \begin{bmatrix} - \\ - \\ \dots \\ + \\ + \end{bmatrix} = \begin{bmatrix} + \\ + \\ \dots \\ + \\ + \end{bmatrix} \end{aligned}$$

In the last expression, each $+$ is a positive number. Note that elements of $\mathbf{X}$ cannot be zero because of our assumption that each column of W has at least one non-zero entry. Now define, $\mathbf{r} = -W\mathbf{X}$ and we must show that $\mathbf{r}$ has at least one negative entry. Compute

$$\mathbf{r} = -W\mathbf{X} = -W^{-1}Wv = v.$$

Therefore, $\mathbf{r}$ has at least one negative entry under our assumption that W has at least one column with non-negative entries.

Note that our proof actually gives infinitely many $\mathbf{X}$ that satisfy the theorem, one for each v having the sign pattern defined in the proof. Moreover, there may exist additional $\mathbf{X}$ that are different from the ones generated by our proof.

While it is possible that cortical circuits somehow restrict themselves to the subsets of parameter space that maintain a positive solution to Eq. 3.4 across all salient stimuli, we consider the alternative hypothesis that Eq. 3.4 and the balanced network theory that underlies it do not capture the full spectrum of cortical circuit dynamics.

3.4.2 Semi-balanced State in BRNNs

To explore spiking network dynamics when Eq. 3.4 predicts negative rates, we considered the same network as above but changed the feedforward connection probabilities so that Eq. 3.4 predicts positive firing rates only when r_{x1} and r_{x2} are nearly equal. When r_{x2} is much larger than r_{x1}, Eq. 3.4 predicts negative firing rates for population e_1, and vice versa, due to a **competitive dynamic**.

Simulating the network with $r_{x1} = r_{x2}$ produces positive rates, asynchronous-irregular spiking, and excitatory-inhibitory balance in Figure 3.1d, the first 500ms. Increasing r_{x2} to where Eq. 3.4 predicts negative rates for population $e1$ causes spiking to cease in $e1$ due to an excess of inhibition in Figure 3.1d, the last 500ms.

Notably, however, input currents to populations $e2$ and i remain balanced when $e1$ is silenced (see Figure 3.1c) so the i and $e2$ populations form a balanced subnetwork. These simulations demonstrate a network state that is not balanced in the classical sense because one population receives excess inhibition. However,

1. no population receives excess excitation,
2. any population with excess inhibition is silenced, and
3. the remaining populations form a balanced sub-network.

Here, an excess of excitation (inhibition) in population a should be interpreted as $\mathbf{I}_a \sim \mathcal{O}(\overline{JK})$ with $\mathbf{I}_a > 0$ ($\mathbf{I}_a < 0$). The three conditions above can be re-written mathematically in the large JK limit as two conditions,

1. $[W\mathbf{r} + \mathbf{X}]_a \leq 0$ for all populations a, and
2. If $[W\mathbf{r} + \mathbf{X}]_a < 0$, then $\mathbf{r}_a = 0$.

These conditions, along with the implicit assumption that $\mathbf{r} \geq 0$, define a generalization of the balanced state. We refer to networks satisfying these conditions as **"semi-balanced"** since they require that strong excitation is canceled by inhibition, but they do not require that inhibition is similarly canceled. Note that the condition $[W\mathbf{r} + \mathbf{X}]_a \leq 0$ does not mean $\mathbf{I}_a < 0$, but only that $\mathbf{I}_a \sim \mathcal{O}(1)$ whenever $\mathbf{I}_a \geq 0$ so that $[W\mathbf{r} + \mathbf{X}]_a = 0$ in the large $\overline{JK}$ limit, *i.e.*, no excess excitation.

In other words, populations in the semi-balanced state can receive $\mathcal{O}(\overline{JK})$ net-inhibitory input, but if their input is net-excitatory, it must be $\mathcal{O}(1)$. Hence, the semi-balanced state is characterized by excess inhibition, but not excess excitation, to some neural populations. In contrast, the balanced state requires net input to be $\mathcal{O}(1)$ regardless of whether it is net-excitatory or net-inhibitory, hence no excess excitation or inhibition. Note that firing rates remain $\mathcal{O}(1)$ in both the balanced and semi-balanced states.

How are firing rates related to connectivity in semi-balanced networks? we prove the **Theorem 2** below that semi-balanced networks satisfy

$$\mathbf{r} = [W\mathbf{r} + \mathbf{X} + \mathbf{r}]^{+} \tag{3.5}$$

in the limit of large $\overline{JK}$, where $[x]^{+} = max(0, x)$ is the positive part of x (a.k.s rectified linear or threshold-linear) function. Eq. 3.5 generalized Eq. 3.4 to allow for excess inhibition.

We now prove that Eq. 3.5, which specifies firing rates in the semi-balanced state is equivalent to the two conditions preceding it, which define the semi-balanced state.

Theorem 2 *Suppose W is an $n \times n$ matrix and $\boldsymbol{X}$ is an $n \times 1$ vector. An $n \times 1$ vector $\boldsymbol{r}$ satisfies A)*

$$[W\boldsymbol{r} + \boldsymbol{X} + \boldsymbol{r}]^{+} = \boldsymbol{r}$$

if and only if it satisfies the following three conditions at every index $a = 1, \ldots, n$:

1. $[W\mathbf{r} + \mathbf{X}]_a \leq 0$
2. If $[W\mathbf{r} + \mathbf{X}]_a = 0$, then $\mathbf{r}_a = 0$
3. $\mathbf{r}_a \geq 0$.

Proof. We first show that A implies conditions 1–3. Assume $\mathbf{r}$ satisfies A and consider some index, a. We need to show that 1–3 is all satisfied at a. Condition 3 is satisfied because $\mathbf{r}_a = [\ldots]^{+} \geq 0$. We still need to prove that conditions 1–2 are satisfied. Note that we either have $\mathbf{r}_a = 0$ or $\mathbf{r}_a \geq 0$. First, consider the case that $\mathbf{r}_a = 0$. Then 2 is satisfied automatically and we only need to prove 1. If $\mathbf{r}_a = 0$ then, by A, $[W\mathbf{r} + \mathbf{X}]_a^{+} = \mathbf{r}_a = 0$ which implies that $[W\mathbf{r} + \mathbf{X}] \leq 0$. Now we must consider the case $\mathbf{r}_a \geq 0$. By A, $[W\mathbf{r} + \mathbf{X}]_a^{+} = \mathbf{r}_a > 0$, so the ReLu is evaluated at its positive part and we can conclude that $\mathbf{r}_a = [W\mathbf{r} + \mathbf{X} + \mathbf{r}]_a = [W\mathbf{r} + \mathbf{X}]_a + \mathbf{r}$. Cancelling the two $\mathbf{r}_a$ terms implies that $[W\mathbf{r} + \mathbf{X}]_a = 0$. Hence, 1 and 2 are both satisfied. This concludes the proof that A implies 1–3.

Now we must prove that 1–3 implies A. We, therefore, assume 1–3 and derive A at each index, a. By 3, we must have $\mathbf{r}_a = 0$ or $\mathbf{r}_a > 0$. First assume $\mathbf{r}_a = 0$. Then $[W\mathbf{r} + \mathbf{X} + \mathbf{r}]_a + = [W\mathbf{r} + \mathbf{X}]_a^{+} = 0$ where the last step follows from our assumption of 1. Therefore, $[W\mathbf{r} + \mathbf{X} + \mathbf{r}]_a + = \mathbf{r}_a = 0$. Now assume $\mathbf{r}_a > 0$. Then, by 1 and 2 combined, we must have $[W\mathbf{r}+\mathbf{X}]_a+ = 0$. Therefore, $[W\mathbf{r}+\mathbf{X}+\mathbf{r}]_a+ = [\mathbf{r}_a]^+ = \mathbf{r}_a$ since $\mathbf{r} > 0$. This completes our proof.

Note that the condition $\mathbf{r}_a \geq 0$ was not explicitly included in the results because it was implicitly assumed. In the first half of our proof, we concluded that $[W\mathbf{r}+\mathbf{X}]_a = 0$ wherever $\mathbf{r}_a > 0$. This implies that balance is maintained at each population that has a non-zero firing rate, *i.e.*, that the populations with non-zero rates form a balanced sub-network.

The equation $[W\mathbf{r}+\mathbf{X}+\mathbf{r}]+ = \mathbf{r}^+$ at first appears awkward because it sums terms with potentially different dimensions: $\mathbf{r}$ has dimension $1/time$ (*e.g.*, units Hz) while $W\mathbf{r}$ and X have dimensions of the neuron model's input current (measured in mV in our model since we normalized by the leak conductance). Even though it is derived in the limit of large $\overline{JK}$. Note that $\mathbf{r}$ satisfies Eq. 3.5 and only if it satisfies $c\mathbf{r} = [W\mathbf{r}+\mathbf{X}+c\mathbf{r}]^+$ for any $c > 0$, which explains why terms with different units can be summed together in Eq. 3.5. The following theorem clarifies that this combination of dimensions is consistent because one can introduce a scaling factor without changing the solution space.

Theorem 3 *Let W be an $n \times n$ matrix and let $\boldsymbol{X}$ and $\boldsymbol{r}$ be $n \times 1$ vectors. The equation 3.5*

$$[W\boldsymbol{r}+\boldsymbol{X}+\boldsymbol{r}]^+ = \boldsymbol{r}$$

is satisfied if and only if the equation

$$[W\boldsymbol{r}+\boldsymbol{X}+c\boldsymbol{r}]^+ = c\boldsymbol{r} \tag{3.6}$$

is satisfied for every $c > 0$.

Proof. We first prove that Eq. 3.5 implies Eq. 3.6. Assume Eq. 3.5 is true. Let a be some index. Either $\mathbf{r}_a = 0$ or $\mathbf{r}_a > 0$. First assume $\mathbf{r}_a = 0$, then $[W\mathbf{r}+\mathbf{X}]_a \leq 0$

and $c\mathbf{r} = 0$. Therefore $[W\mathbf{r} + \mathbf{X} + c\mathbf{r}]_a^+ = [W\mathbf{r} + \mathbf{X}]_a^+ = c\mathbf{r}_a$. Now assume $\mathbf{r}_a > 0$, then $c\mathbf{r}_a > 0$ and as discussed above, we must have $[W\mathbf{r} + \mathbf{X}]_a = 0$. Therefore $[W\mathbf{r} + \mathbf{X} + c\mathbf{r}]_a^+ = [c\mathbf{r}]_a^+ = c\mathbf{r}_a$. This concludes our proof that Eq. 3.5 implies Eq. 3.6.

We must now prove that Eq. 3.6 implies Eq. 3.5. This is trivial because we can simply take $c = 1$.

It is worth noting that the simplest possible semi-balanced network has one inhibitory population and one excitatory population with the excitatory population silenced by the inhibitory population. This would arise when a condition for the positivity of firing rates in a two-population balanced network is violated [39, 83]. Notably, Eq. 3.5 represents a piece-wise linear, but globally **nonlinear** mapping from $\mathbf{X}$ to $\mathbf{r}$.

3.4.3 A Direct Correspondence to ANNs

Unlike balanced networks, semi-balanced networks implement nonlinear stimulus representations in Figure 3.2c. Eq. 3.5 also demonstrates a direct relationship between semi-balanced networks and recurrent artificial neural networks with rectified linear activations used in machine learning [34] and their continuous-time analogues studied by Curto and others under the label "threshold-linear networks" [23, 24, 35, 94]. These networks are defined by equations of the form

$$\tau\dot{\mathbf{r}} = -\mathbf{r} + [U\mathbf{r} + \mathbf{X}]^+. \tag{3.7}$$

Taking $U = W + Id$ where Id is the identity matrix establishes a one-to-one correspondence between solutions to Eq. 3.5 and fixed points of threshold-linear networks or ARNNs. Indeed, we used this correspondence to construct a semi-balanced spiking network that approximates a continuous exclusive-or (XOR) function in Figure 3.2 which is widely known to be impossible with linear networks [34].

Previous work on threshold-linear networks shows that, despite the simplicity of Eq. 3.5, its solution space can be complicated [23, 24, 35, 94]: Any solution is partially specified by the subset of populations, a, at which $\mathbf{r}_a > 0$, called the "support" of the solution. There are 2^n potential supports in a network with n populations, there can be multiple supports that admit solutions, and these solutions can depend in complicated ways on the structure of W and $\mathbf{X}$. Hence, semi-balanced networks give rise to a rich mapping from stimuli, $\mathbf{X}$, to responses, $\mathbf{r}$.

The semi-balanced state is equivalent to bounding rates. Under Eq. 3.3, the semi-balanced state is realized and Eq. 3.5 is satisfied only if firing rates do not grow large as $\overline{JK} \to \infty$ (see **Theorem 4** for proof details). In other words, Eq. 3.5 and the semi-balanced state it describes are general properties of strongly and/or densely coupled networks (large $\overline{JK}$) with moderate firing rates. To the extent that cortical circuits have large $\overline{JK}$ values and moderate firing rates, therefore, Eq. 3.5 provides an accurate approximation to cortical circuit responses.

We now prove that for firing rate models, the semi-balanced state is realized if and only if $\mathbf{r} \sim \mathcal{O}(1)$ as $\overline{JK} \to \infty$. The proof relies on some reasonable assumptions on the f-I curve, *i.e.*, the function $\mathbf{r} = f(\mathbf{I})$.

Theorem 4 *Suppose W is a fixed $n \times n$ matrix and $\boldsymbol{X}$ a fixed $n \times 1$ vector. Assume that $\boldsymbol{r}$ and $\boldsymbol{I}$ are $n \times 1$ vectors that depend on $\overline{JK}$ with equation 3.3*

$$\boldsymbol{I} = \overline{JK}[W\boldsymbol{r} + \boldsymbol{X}]$$

and

$$\boldsymbol{r} = f(\boldsymbol{I})$$

for all sufficiently large values of $\overline{JK} > 0$. Also assume that $f(x)$ is a non-negative, non-decreasing function for which, $\lim_{x\to\infty} f(x) = M$, and $\lim_{x\to-\infty} f(x) = 0$. Here, M can be finite in the case of a saturating or sigmoidal f-I curve, or $M = \infty$ the case *of an f-I curve that does not saturate. If*

$$\boldsymbol{r}^{\infty} = \lim_{\overline{JK}\to\infty} \boldsymbol{r}$$

exists and $\boldsymbol{r}^{\infty} < M$ *for all* $a = 1, ..., n$ *then*

$$[W\boldsymbol{r}^{\infty} + \boldsymbol{X} + \boldsymbol{r}^{\infty}] = \boldsymbol{r}^{\infty} \tag{3.8}$$

Proof. Assume $\lim_{\overline{JK}\to\infty} \mathbf{r}$ exists and is finite. Then we need to show that it satisfies Eq. 3.8. Specifically, for each index, $a = 1, \ldots, n$, we need to show that

$$[[W\mathbf{r}^{\infty} + \mathbf{X}]_a^+ \mathbf{r}_a^{\infty}] = \mathbf{r}_a^{\infty}$$

where $[W\mathbf{r} + \mathbf{X}]_a$ is the ath index of $[W\mathbf{r} + \mathbf{X}]$. Let $a \in [1, \ldots, n]$ be an arbitrary index and define

$$c = \lim_{\overline{JK}\to\infty} \mathbf{I}_a \overline{JK}.$$

Note that

$$c = [W\mathbf{r} + \mathbf{X}]_a = [W\mathbf{r}^{\infty} + \mathbf{X}]_a.$$

exists and is finite by assumption.

We first argue that $c \le 0$. To show this, we will assume that $c > 0$ and prove a contradiction. If $c > 0$, then

$$\lim_{\overline{JK}\to\infty} \mathbf{I}_a = \lim_{\overline{JK}\to\infty} \overline{JK}c = \infty.$$

and therefore

$$\mathbf{r}_a^{\infty} = \lim_{\overline{JK}\to\infty} f(\mathbf{I}_a) = M.$$

which contradicts our assumption that $\mathbf{r}_a^{\infty} < M$ for all M. We may conclude that

$c \leq 0$. We now break the proof into two cases: $c = 0$ and $c < 0$. *Case I:* $c = 0$. We have $c = [W\mathbf{r} + \mathbf{X}]_a = 0$, so

$$[[W\mathbf{r}^\infty + \mathbf{X}]_a + \mathbf{r}_a^\infty] = [\mathbf{r}_a^\infty]^+$$

but $\mathbf{r}_a^\infty > 0$ at all indices, a, because $\mathbf{r} = f(\mathbf{I}) \geq 0$ at all $\overline{JK}$ and $\mathbf{r}^\infty = \lim_{\overline{JK}\to\infty} \mathbf{r}$. Therefore,

$$[[W\mathbf{r}^\infty + \mathbf{X}]_a + \mathbf{r}_a^\infty] = [\mathbf{r}_a^\infty]^+ = \mathbf{r}_a^\infty.$$

This completes Case 1. *Case I:* $c < 0$. We have $c = [W\mathbf{r} + \mathbf{X}]_a < 0$, so

$$\lim_{\overline{JK}\to\infty} \mathbf{I}_a = \lim_{\overline{JK}\to\infty} \overline{JK}c = -\infty.$$

Therefore,

$$\mathbf{r}_a^\infty = \lim_{\overline{JK}\to\infty} f(\mathbf{I}_a) = \lim_{\mathbf{I}_a\to\infty} f(\mathbf{I}_a) = 0.$$

As a result,

$$[[W\mathbf{r}^\infty + \mathbf{X}]_a + \mathbf{r}_a^\infty]^+ = [[W\mathbf{r}^\infty + \mathbf{X}]_a]^+ = 0 = \mathbf{r}_a^\infty.$$

because $[W\mathbf{r} + \mathbf{X}]_a = c < 0$ and $\mathbf{r}_a^\infty$. This completes Case 2.

In summary, our results establish a direct mapping from biologically realistic cortical circuit models to recurrent artificial neural networks used in machine learning and to the rich mathematical theory of threshold-linear networks.

3.4.4 Homeostatic Plasticity Produces "Detailed Semi-balanced"

So far, we have only considered firing rates and excitatory-inhibitory balance averaged over neural populations. Cortical circuits implement distributed neural representations that are not always captured by homogeneous population averages [75]. Balance realized at the single-neuron resolution, **i.e.**, where the input to each neuron is balanced, is often referred to as "detailed balance" [40, 84]. We, therefore, use the term "**detailed semi-balance**" for semi-balance realized at single neuron resolution.

Specifically, generalizing the definitions of population-level balance and semi-balance above, detailed balance is defined by requiring that the net synaptic input to all neurons is $\mathcal{O}(1)$. Detailed semi-balanced only requires neurons' input to be $\mathcal{O}(1)$ when it is net-excitatory. Net-inhibitory input to some neurons will be $\mathcal{O}(\overline{JK})$ in the detailed semi-balanced state. As such, the distribution of total synaptic input to neurons in the semi-balanced state will be left-skewed, indicating strong inhibition to some neurons, but no comparably strong excitation.

To explore detailed balance and semi-balance, we first considered the same spiking network considered above, but with only a single excitatory, inhibitory, and external population (Figure 3.3a). To model a stimulus with a distributed representation, we first added an extra external input perturbation that is constant in time but randomly distributed across neurons. Specifically, the time-averaged synaptic input to each neuron was given by the $N \times 1$ vector

$$\vec{I} = -\overline{JK}[J\vec{r} + \vec{X}] \tag{3.9}$$

where J is the $N \times N$ recurrent connectivity matrix and $\vec{r}$ is the $N \times 1$ vector of firing rates. Note that we use the arrow notation, $vecI$, for N-dimensional vectors to distinguish them from boldfaced mean-field vectors, like $\mathbf{I}$, that have dimensions equal to the number of populations.

We apply the same notational convention to $\vec{r}, \vec{X}$, etc. For a given $\vec{Z}$ the mean N-dimensional external input to each neuron is given by

$$\vec{X} = J_x\vec{r}_x + \vec{Z}$$

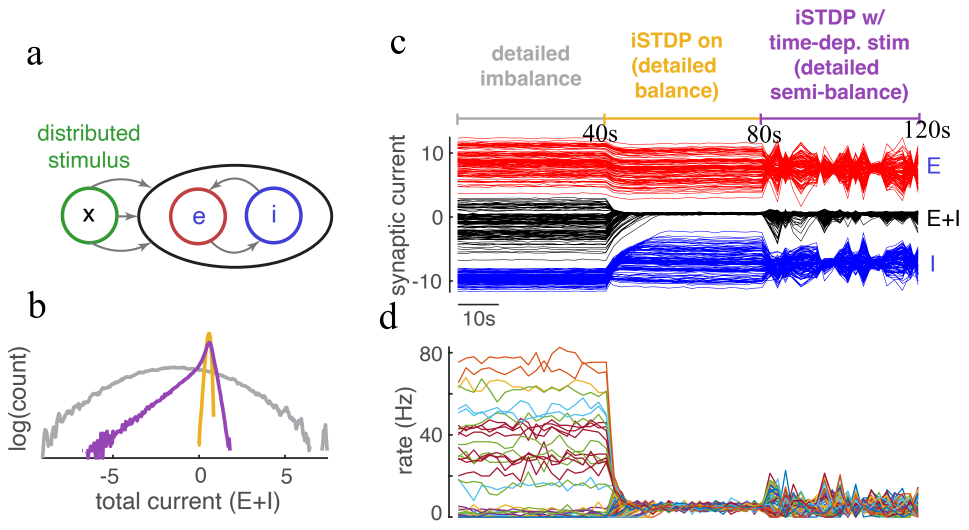

Figure 3.3. **Detailed imbalance, balance, semi-balance, and distributed neural representations** a: Network diagram. Same as in Figure 3.1a except there is just one excitatory and one external population and an additional input $\vec{Z} = \sigma_1 \vec{Z}_1 + \sigma_2 \vec{Z}_2$. b: Histograms of input currents to all excitatory neurons averaged over the first 40s (gray, imbalanced), the next 40s (yellow, balanced), and the last 40s (purple, semi-balanced). c: Excitatory (red), inhibitory (blue), and total (black) input currents to 100 randomly selected excitatory neurons averaged over 2s time bins. During the first 40s, synaptic weights and $\sigma_1 = \sigma_2$ were fixed. During the next 40s, homeostatic iSTDP was turned on and $\sigma_1 = \sigma_2$ were fixed. During the last 40s, iSTDP was on and σ_1 and σ_2 were selected randomly every 2s. d: Firing rates of the same 100 neurons averaged over 2s bins.

where, J_x and $\vec{r}_x$ are the feedforward connectivity matrix and external rates. The distributed stimulus, $\vec{Z}$, is defined by

$$\vec{Z} = \sigma_1 \vec{Z}_1 + \sigma_2 \vec{Z}_2$$

where $\vec{Z}_1$ and $\vec{Z}_2$ are standard normally distributed, N $\times$ 1 vectors. The vector, $\vec{Z}$, lives on a two-dimensional hyperplane in N-dimensional space parameterized by σ_1 and σ_2. Hence, $\vec{Z}$ models a two-dimensional stimulus whose representation is distributed randomly across the neural population.

Since we are primarily interested in the encoding of the perturbation, $\vec{Z}$, we could have replaced the spike-based, Poisson synaptic input from the external population with a time-constant, DC input to each neuron as in previous work [82]. We chose to keep the spike-based input to add biological realism and to demonstrate the encoding of $\vec{Z}$ is robust to the Poisson noise induced by the background spike-based input. A more biologically realistic model might encode $\vec{Z}$ in the spike times themselves instead of using an additive perturbation.

Simulations show that this network does not achieve detailed balance or semi-balance: Some neurons receive excess inhibition and some receive excess excitation (Figure 3.3c, first 40s), leading to large firing rates in some neurons (Figure 3.3b) and broad distribution of total input currents (Figure 3.3b). Indeed, it has been argued previously that randomly connected networks break detailed balance when stimuli and connectivity are not co-tuned [40, 52]. This is consistent with previous results on "imbalanced amplification" in which connectivity matrices with small-magnitude eigenvalues values can break balance when external inputs are not orthogonal to the corresponding eigenvectors [29]. When J is large and random, it will have many eigenvalues near the origin, which can lead to imbalanced amplification if $\vec{X}$ is not orthogonal to the corresponding eigenvectors.

Previous work shows that detailed balance can be realized by a homeostatic, **inhibitory spike-timing dependent plasticity (iSTDP)** rule [40, 84]. Indeed, when iSTDP was introduced in our simulations, the detailed balance was obtained and firing rates became more homogeneous (Figure 3.3c and d, the second 40s) with a much narrower distribution of total input currents (Figure 3.3b, yellow), indicating detailed balance, at least while σ_1 and σ_2 were held fixed.

Of course, real cortical circuits receive time-varying stimuli. To simulate time-varying stimuli, we randomly selected new values of σ_1 and σ_2 every 2s (Figure 3.3c and d, the last 40s). This change lead to some neurons receiving excess inhibition in response to some stimuli, but neurons did not receive correspondingly strong excess excitation (Figure 3.3c, black curves the last 40s) resulting in a left-skewed distribution of synaptic inputs (Figure 3.3b, purple). These results are consistent with a detailed semi-balanced state, which is characterized by excess inhibition to some neurons, but a lack of similarly strong excitation. These results show that detailed semi-balance, but not detailed balance, is naturally achieved in circuits with iSTDP and time-varying stimuli.

To gain a better intuition for why the distribution in (Figure 3.3b, purple) is left-skewed, consider the network with iSTDP and time-varying stimuli. iSTDP changes weights in a way that encourages all excitatory firing rates to be close to a target rate [84] (we used a target rate of 5Hz). In the presence of a stimulus that varies faster than the iSTDP learning rate, the network cannot achieve the target rates for every neuron at every stimulus. However, the network is pushed strongly away from states with large, net-excitatory input to some neurons because those states produce large firing rates that are very far from the target rates. On the other hand, the network is not pushed as strongly away from states with large net-inhibitory input to some neurons because those states produce firing rates of zero for those neurons, which is not so far from the target rates.

Repeating our simulations in a model with conductance-based synapses shows that shunting inhibition prevents strong inhibitory currents, consistent with evidence that shunting inhibition is prevalent in visual cortex [15], but if currents are measured under voltage clamp then recorded currents are similar to those in Figure 3.3b-d, with excess hyperpolarizing currents in the semi-balanced state.

Firing rates in the detailed semi-balanced state are not very broadly distributed (Figure 3.3c, the last 40s), which is inconsistent with some cortical recordings. Note that the broadness of the firing rate distribution is partly a function of the magnitude of the perturbation strengths, σ_1 and σ_2. Also, all of our perturbations lie on a two-dimensional plane, so they could potentially be balanced more effectively by iSTDP than higher-dimensional perturbations. Finally, our iSTDP rule used the same target rate for all neurons, which may not be realistic. Stronger perturbations, higher-dimensional perturbations, and variability in target rates, among other factors, could lead to broader firing rate distributions in the detailed semi-balanced state. The width of firing rate distributions for naturalistic stimuli should be considered in future work but is outside the scope of this study.

In summary, when networks are presented with time-varying stimuli, iSTDP produces a detailed semi-balance, but not a detailed balance.

3.5 Discussion

We introduced the semi-balanced state, defined by an excess of inhibition without an excess of excitation. This state is realized naturally in networks for which the classical balanced state cannot be achieved and produces nonlinear stimulus representations, which are not possible in classical balanced networks. We established a direct mathematical relationship between semi-balanced networks, artificial neural networks, and the rich mathematical theory of threshold-linear networks. Detailed semi-balance is realized naturally in networks with iSTDP and time-varying stimuli and produces nonlinear stimulus representations that improve the network's computational properties.

Previous work revealed multi-stability and nonlinear transformations at the level of population averages by balanced networks with short-term synaptic plasticity [60]. Future work should consider how the nonlinearities introduced by short-term plasticity combine with the nonlinearities introduced by semi-balance.

Classical balanced networks are balanced at the population level, but not necessarily at the level of individual neurons (no detailed balance). While such networks can only perform linear computations at the level of population averages, they can perform nonlinear computations at the level of single neurons and their firing rate fluctuations [21, 50, 51]. Cortical circuits do appear to perform nonlinear computations at the population level. For example, population responses to high-contrast visual stimuli add sub-linearly, which can be captured by supralinear stabilized networks (SSNs) [74] and semi-balanced networks.

We demonstrated that semi-balanced networks can implement a continuous XOR nonlinearity at the population level (Figure 3.2c) and detailed semi-balanced networks implement more intricate nonlinearities at the resolution of single neurons (Figure 3.3b and c), but we did not consider additional types of nonlinearities. Future work should more completely explore the types of nonlinearities that can be expressed by solutions to Eq. 3.5.

We showed that networks with iSTDP achieve detailed semi-balance and produce nonlinear representations at the level of individual neurons (Figure 3.3c and d). However, we do not mean to suggest that iSTDP or balance is responsible for the presence of nonlinear representations. iSTDP is needed for achieving detailed semi-balance, not nonlinear representations. However, networks without iSTDP are imbalanced at the resolution of individual neurons (detailed imbalance, see Figure 3.3b-d, gray). In summary, our results show that networks with iSTDP can produce a form of detailed balance (detailed semi-balance) while still implementing nonlinear representations.

One limitation of our approach is that it focused on fixed point rates and did not consider their stability or the dynamics around fixed points. Indeed, fluctuations of firing rates and total synaptic inputs are $\mathcal{O}(1)$ under the scaling of synaptic weights that we used. When a solution to Eq. 3.5 exists, it represents a fixed point of Eq. 3.2 in the $\overline{JK} \to \infty$ limit. The fixed point is stable when all eigenvalues of the Jacobian matrix of Eq. 3.5 evaluated at the fixed point have a negative real part. Previous work shows that balanced networks can exhibit spontaneous transitions between attractor states [57] which can be formed by iSTDP [57, 84]. Attractor states in those studies maintained strictly positive firing rates across populations, keeping the networks in the classical balanced state. This raises the question of whether similar attractors could arise in which some populations are silenced by excess inhibition, putting them in a semi-balanced state. Tools for studying these states, and for studying stability and dynamics more generally, could potentially be developed from the mathematical theory of threshold-linear networks [23, 24, 35, 94].

The semi-balanced state is defined by an excess of inhibition without a corresponding excess of excitation. This is at first glance consistent with evidence that inhibition dominates cortical responses in awake animals [37]. However, it should be noted that synaptic conductances, not currents, were reported and they only reported conductances relative to their peaks, not raw conductances [37]. It is therefore difficult to draw a direct relationship between the results in [37] to our results on balance or semi-balance. In addition, if synaptic currents are measured under a voltage clamp with the potential clamped sufficiently far between the excitatory and inhibitory reversal potentials, we predict a skewed distribution of currents with a heavier tail of hyperpolarizing versus depolarizing currents (Figure 3.3b, purple). These predictions should be tested more directly using *in vivo* recordings.

The relationship between connectivity and firing rates in recurrent spiking networks can be mathematically difficult to derive, which can make it difficult to derive gradient-based methods for training recurrent spiking networks (though some studies have succeeded, see for example [48, 62]). The piece-wise linearity of firing rates in the semi-balanced state, Eq. 3.5 could simplify the training of recurrent spiking networks because the gradient of the firing rate with respect to the weights can be easily computed. This could have implications for the design and training of connectivity in neuromorphic hardware.

In summary, semi-balanced networks are more biologically parsimonious and computationally powerful than widely studied balanced network models. The foundations of semi-balanced network theory presented here open the door to several directions for further research.

CHAPTER 4

CAN HOMEOSTATIC PLASTICITY LEARN TO COMPUTE PREDICTION ERRORS?

This chapter is adapted from [96].

In the previous chapter, we discovered the properties of strongly connected BRNNs under different stimulus representations. We extended the balanced theory into a semi-balanced state and demonstrated that homeostatic inhibitory plasticity can achieve semi-balanced states in a detailed individual neuron level with time-varying stimuli. These findings lay a foundation for us to understand more functionalities that the brain can "do". One of such is that the brain is believed to make predictions about sensory stimuli and encode deviations from these predictions in the activity of "prediction error neurons." For example, in the visuomotor system, head movements produce predictable flows of an animal's visual scene. Visual cortical circuits learn predictable associations between bottom-up input from the visual stream and top-down input from the motor system. Violations of the learned predictions, known as "mismatched stimuli" or "**prediction errors**", produce distinct responses in visual cortical neurons, which can help the animal distinguish between self-driven and externally driven movements of its visual scene [8, 43, 47, 53].

The principle that cortical neuronal networks can make predictions about sensory stimuli and detect errors in these predictions defines the widely influential theory of **predictive coding**. Although the precise circuitry and learning mechanisms through which the brain can learn to compute and update its predictions are unknown, homeostatic inhibitory synaptic plasticity is a promising mechanism for training neuronal networks to perform predictive coding. **Homeostatic plasticity** causes neurons to maintain a steady, baseline firing rate in response to inputs that closely match the inputs on which a network was trained, but firing rates can deviate away from this baseline in response to stimuli that are mismatched from training.

We combine computer simulations and mathematical analysis systematically to test the extent to which randomly connected, unstructured networks can learn to compute prediction errors through homeostatic inhibitory synaptic plasticity. We find that homeostatic plasticity alone is sufficient for learning prediction errors for trivial time-constant stimuli, but not for more realistic time-varying stimuli. We use a mean-field theory of plastic networks to explain our findings.

4.1 Introduction

The idea that the brain uses predictions and prediction errors to encode and interpret sensory information dates back to 19th century work by Helmholz [46, 87] and underlies more general theories of neural function such as predictive coding, predictive processing, active inference, and the free energy principle [22, 31, 46, 69]. The question of how neural circuits compute prediction errors and how they learn predictions through biologically plausible synaptic plasticity rules is not settled, but some theories have been put forward [12, 14, 42, 70, 76, 88, 90].

Cortical neurons are highly interconnected, even within a single cortical area and layer. This dense, recurrent, and intralaminar connectivity shapes the intrinsic dynamics and stimulus responses of local cortical circuits. The nonlinear firing rate dynamics that arise from this recurrent connectivity can interact with the slower dynamics of synaptic plasticity in complex ways. Homeostatic inhibitory synaptic plasticity is a widely observed and widely studied type of synaptic plasticity [19, 20, 40, 58, 76, 85, 86] in which the strength of inhibitory synapses are adjusted in an activity-dependent manner that tends to push the postsynaptic neurons' firing rates toward a homeostatic baseline target. Simulations and theoretical analyses of mathematical models of homeostatic inhibitory plasticity show that, while firing rates are near their targets in response to stimuli on which the network has been trained, firing rates deviate from their targets in response to unfamiliar stimuli in these models [3, 9, 41, 42, 76, 85].

As in related computational work [41, 42, 76] 2021), we conjectured that homeostatic inhibitory plasticity could learn to perform some type of predictive coding. In particular, if the external input to a neural population were formed from bottom-up and top-down stimuli, then homeostatic plasticity in the network would naturally learn to produce baseline activity in response to "matched" top-down and bottom-up pairings (*i.e.*, pairings that are similar to those on which the network was trained). On the other hand, "mismatched" pairings (*i.e.*, pairings from outside the training distribution) would produce firing rate responses that are further from the homeostatic baseline. In this sense, the network should learn to encode prediction errors (*i.e.*, errors in the ability to predict top-down input from bottom-up input or *vice versa*) in the deviation of the firing rates from their baseline. Importantly, and in contrast to previous work [41, 42, 76], we conjectured that the network should not need to be imparted with any special structure or architecture to learn this computation since homeostatic plasticity should naturally achieve this result due to its tendency to produce baseline responses to stimuli on which the network was trained, but not in response to novel stimuli.

To test our conjecture, we used an unstructured, recurrent, spiking neuronal network model endowed with a homeostatic inhibitory plasticity rule receiving two sources of external input, modeling top-down and bottom-up stimuli. We trained the network with given patterns of top-down and bottom-up pairings, interpreted as "matched" stimuli, before presenting a "mismatched" stimulus that deviated from the pairings used during training. Numerical simulations showed that the network reliably produced baseline firing rates for a fixed pair of bottom-up and top-down inputs during training, and deviated from baseline in response to a mismatched stimulus. A mean-field firing rate model and a mathematical analysis using a separation of timescales helped reveal the dynamics underlying these numerical simulations. Hence, homeostatic plasticity learns to compute prediction errors whenever top-down and bottom-up stimuli are fixed during training. However, useful predictive coding algorithms should learn to detect relationships between time-varying top-down and bottom-up inputs. We generalized our input model to vary the intensity of top-down and bottom-up inputs in unison. An effective learning algorithm should learn to detect a prediction error whenever the intensity changes out of unison. To our surprise, our spiking network with homeostatic synaptic plasticity was unable to learn to detect this type of prediction error, even in a relatively simple (time-varying) setting. Going back to our mean-field analysis helped to clarify how and why the model failed to perform predictive coding in this setting after succeeding in the simpler (time-constant) setting.

We conclude that homeostatic inhibitory synaptic plasticity alone is not sufficient to learn and perform non-trivial predictive coding in unstructured neuronal network models. Previous theoretical work shows that network models that carefully account for the connectivity structure of multiple inhibitory subtypes are able to learn prediction errors using homeostatic plasticity, even for inputs where top-down and bottom-up input co-vary in intensity [41, 42]2021). Hence, the failure of our model in this scenario implies that network structure is critical for successfully learning predictive coding tasks with homeostatic plasticity.

4.2 Model Descriptions

4.2.1 An EIF Network Model with Homeostatic Plasticity

We first consider a computational model of a local cortical circuit composed of $N = 5000$ randomly connected exponential integrate-and-fire (EIF) spiking neuron models ($N_e = 4000$ of which are excitatory and $N_i = 1000$ inhibitory) [16, 32]. The membrane potentials of neuron j in population $a = e, i$ obeys

$$\tau_m \frac{dV_j^a}{dt} = -(V_j^a - E_L) + D_T e^{\frac{V_j^a - V_T}{D_T}} + I_j^a(t) \tag{4.1}$$

with the added condition that each time $V_k(t)$ crosses a threshold at V_{th} , it is reset to V_{re} and a spike is recorded. The synaptic input to neuron j in population a is modeled by

$$I_j^a(t) = X_j^a(t) + \sum_{b=e,i} \sum_{k=1}^{N} J_{jk}^{ab} \alpha_b(t - t_{n,k}^b)$$

where $X_j^a(t)$ models external synaptic input, $J_{jk}^{ab}$is a synaptic weight, $t_{n,k}^b$ is the time of the nth spike of neuron k in population b, and $\alpha_b(t) = \frac{e^{-\frac{t}{\tau_b}}}{\tau_b} H(t)$ is a synaptic filter with $H(t)$ the Heaviside step function.

Initial connectivity in the model is random (connection probability $p = 0.1$) with initial weights, J_{jk}^{ab}, determined only by pre- and post-synaptic neuron type ($J_{jk}^{ab} = j_{ab}$ for connected neurons). Excitatory connectivity, J_{jk}^{ae}, remained fixed, but inhibitory connectivity evolves according to a homeostatic, inhibitory spike-timing-dependent plasticity (iSTDP) rule [3, 40, 85]. Specifically, each time that neuron j in population $a = e, i$ spikes (which occurs at times $t_{n,j}^a$), the inhibitory synaptic weights targeting that neuron are updated according to

$$J_{jk}^{ai} = J_{jk}^{ai} - \eta_a x_k^i(t_{j,n}^a)$$

where a is a learning rate and recall that $t^a_{j,n}$ is the time of the nth spike of neuron j in population $a = e, i$. Additionally, each time inhibitory neuron k spikes, its outgoing synaptic weights are updated according to

$$J^{ai}_{jk} = J^{ai}_{jk} - \eta_a x^a_k(t^i_{j,n} - 2r^a_0)$$

where $t^i_{k,n}$ is the time of the nth spike of inhibitory neuron k. The time series, $x^a_j(t)$ are defined by the differential equation

$$\tau_{STDP}\frac{dx^a_j}{dt} = -x^a_j$$

in addition to the rule that $x^a_j(t)$ is incremented each time that neuron j in population $a = e, i$ spikes according to,

$$dx^a_j(t^a_{j,n}) \leftarrow dx^a_j(t^a_{j,n}) + \frac{1}{\tau_{STDP}} \tag{4.2}$$

As a result, $x^a_j(t)$ estimates the firing rate of neuron j in population a by performing an exponentially-weighted sliding average of the spike density. This plasticity rule tends to push excitatory and inhibitory firing rates toward their target rates, r^e_0 and r^i_0, respectively (see [3, 9, 40, 85] and the mean-field theory presented below).

4.2.2 Simulation Parameters

All simulations were performed by numerically solving the corresponding differential equations using the forward Euler method in custom-written Python code.

For spiking network simulations (Eqs. (4.1)–(4.2); Figures 4.1, 4.4, and 4.5) and mean-field rate network simulations (Eqs. (4.6)–(4.7); Figures 4.2 and 4.6) we used a time step size of $dt = 0.1$ms. For the slow-timescale model (Eqs. (4.24); Figures 4.3 and 4.7) we used a time step size of $dt = 1$s.

4.2.2.1 In the Spiking Network Model

For all spiking network simulations (Eqs. (4.1)–(4.2); Figures 4.1, 4.4, and 4.5), we used $N_e = 4000$ and $N_i = 1000$ excitatory and inhibitory neurons. All neurons were connected with probability $p_{ee} = p_{ei} = p_{ie} = p_{ii} = 0.1$. Connected neurons had initial synaptic weights $j_{ee} = 7.07$mV/ms, $j_{ei} = -49.5$mV/ms, $j_{ie} = 31.8$mV/ms, and $j_{ii} = -70.7$mV/ms. EIF neuron parameters were $\tau_m = 15$ms, $E_L = -72$mV, $V_{re} = -73$mV, $D_T = 2$mV, $V_T = -55$mV, $V_{th} = 0$mV, and a reflecting lower boundary on the membrane potential was placed at $V_{lb} = -80$mV to approximate an inhibitory reversal potential. Synaptic timescales were $\tau_e = 6$ms and $\tau_i = 4$ms. Baseline external input to excitatory and inhibitory neurons was $X_e^0 = 42.4$mV and $X_i^0 = 28.3$mV. Parameters for the inhibitory plasticity rule were $\eta_e = 56.6$mV, $\eta_i = 28.3$mV, and $\tau_{STDP} = 200$ms with target rates at $r_0^e = 4$Hz and $r_0^i = 8$Hz.

4.2.2.2 In the Mean-Field Rate Network Model

For mean-field rate network simulations (Eqs. (4.6)–(4.7); Figures 4.2 and 4.6), we used a gain of $g = 0.001$ms/mV, which was derived by simulating the spiking network model without plasticity and then fitting the f-I curve $r = f(I) = gIH(I)$ (where H is the Heaviside step function) to the time-averaged firing rates and input currents of all neurons in the simulation. Learning rates for rate network simulations were $\eta_e = 8944$mV and $\eta_i = 4472$mV. All other parameters were the same as those used in spiking network simulations.

We are interested in understanding the extent to which such networks can learn to perform predictive coding [14, 46, 69]. More specifically, we reasoned that neurons would spike close to their target rates in response to stimulus patterns similar to those on which they were trained, but deviate from the target rates in response to stimuli that deviate from the training stimuli. In other words, the deviation of firing rates from their targets should encode a "prediction error," *i.e.*, a deviation of the inputs from the patterns that appeared during training.

4.3 Detectable Prediction Errors After Training

4.3.1 Time-constant Inputs

For illustrative purposes, we first considered a simple input model for which the excitatory population was divided into two sub-populations, e_1 and e_2, with $N_{e_1} = N_{e_2} = 2000$ neurons in each sub-population (Figure 4.1A, B). Recurrent connectivity did not depend on sub-population membership, so the network was completely unstructured. During training, each neuron in populations e_1 and e_2 received external stimuli of the form (Figure 4.1A)

$$\left.\begin{aligned} X_{e_1} &= X_e^0 + U \\ X_{e_2} &= X_e^0 + V \end{aligned}\right\} \text{matched} \tag{4.3}$$

where X_e^0 is a baseline input that assures neurons spike at reasonable rates, U is a perturbation modeling bottom-up input, and V is a perturbation modeling top-down input. We used positive bottom-up input and negative top-down input,

$$\begin{aligned} U &= X_e^0/5 \\ V &= -X_e^0/5, \end{aligned} \tag{4.4}$$

but our results are not sensitive to this specific choice of inputs. We refer to this as a "matched" stimulus because it defines the matching of bottom-up with top-down stimuli that the network is trained on. After training on matched stimuli, we modeled mismatched stimuli by the absence of top-down input (Figure 4.1B),

$$\left.\begin{aligned} X_{e_1} &= X_e^0 + U \\ X_{e_2} &= X_e^0 \end{aligned}\right\} \text{mismatched.} \tag{4.5}$$

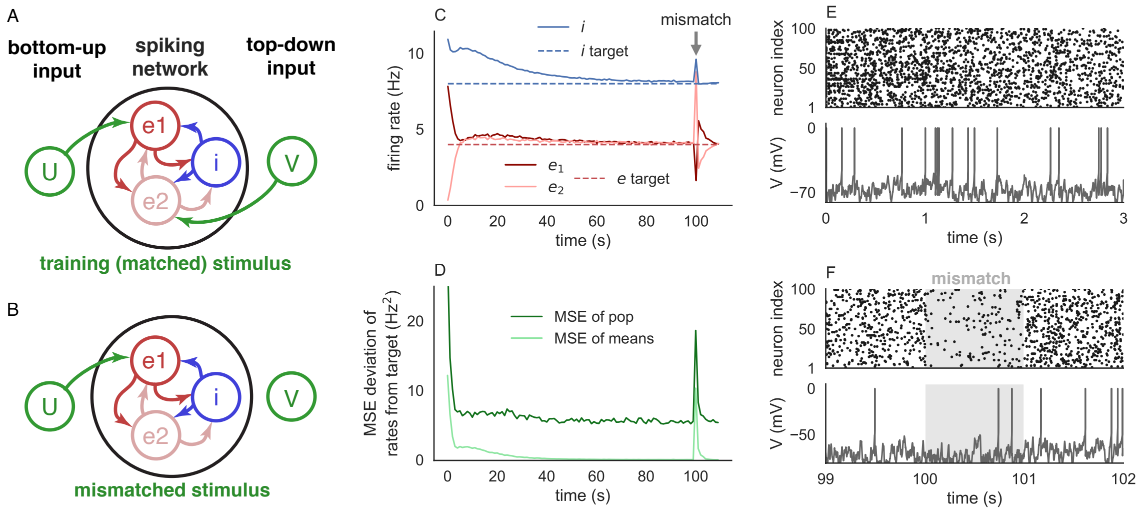

Figure 4.1. **Prediction errors after training on time-constant inputs to multiple sub-populations.** A and B: Network diagram with "training" and "mismatch" stimuli respectively. A randomly connected, recurrent spiking neural network of $N = 5000$ neurons consisted of two excitatory sub-populations (e_1 and e_2) and one inhibitory (i) population. During the first 100s of the simulation, the network received a "training" stimulus in which e_1 and e_2 received extra external input modeling bottom-up and top-down stimuli respectively (A). Then a "mismatch" stimulus was introduced for 1s by removing the top-down stimulus to population e_2. C: Homeostatic inhibitory synaptic plasticity caused population-averaged firing rates to converge to their targets during training, but they deviated from their targets in response to the mismatch stimulus. D: The deviation of the mean firing rates from their targets (MSE_{mean}) and the mean deviation of individual neurons' firing rates (MSE_{pop}) quantify the deviation of firing rates from their targets. E and F: Raster plots (top) and membrane potential (bottom) of a random subset of neurons from population e_1.

We refer to these stimuli as “mismatched” because the top-down and bottom-up inputs are mismatched when compared to the “matched” pairings used to train the network. Mismatched stimuli could also be modeled by an absence of bottom-up input, or any other deviation from the inputs used for training.

We hypothesized that, after training on matched stimuli, the network would produce firing rates close to the target rates in response to matched stimuli and produce firing rates further from the target rates in response to mismatched stimuli.

At the beginning of the simulation mean excitatory and inhibitory firing rates deviated from their targets, but inhibitory plasticity pushed them toward their targets over the course of tens of seconds (Figure 4.1C). After 100s of training on matched stimuli, we tested a mismatched stimulus for 1s. Consistent with our hypothesis, the mean firing rates of each population were further from their targets in response to the mismatched stimulus (Figure 4.1C).

4.3.2 An Analysis of Multiple Sub-populations for the Error Detection

We quantified the distance of the firing rates from their targets from spiking network simulations using two methods. For the first method, we computed the MSE of the **population-averaged** firing rates (Figure 4.1D, light green),

$$MSE_{mean} = \sum_{a=e_1,e_2,i} q_a(r_a - r_a^0)^2$$

where r_a^0 is the target rate and r_a is the mean firing rate of each population averaged over neurons in that population and averaged over time windows of size $T = 1$s. The coefficients $q_a = N_a/N$ represent the proportion of the network contained in each population ($q_{e_1} = q_{e_2} = 0.4$ and $q_i = 0.2$ for our network). Hence, MSE_{mean} weights the errors of larger sub-populations more heavily.

The MSE_{mean} measures how far the population-average rates differ from their target rates but does not measure the deviation of individual neurons' firing rates. Despite the fact that external input was constant across time and the simulations were deterministic (with the exception of "quenched" randomness from the random connectivity), neurons exhibited substantial variability in their spike timing and membrane potential dynamics (Figure 4.1E,F). These dynamics are characteristic of an asynchronous-irregular state [7, 17, 18, 21, 71, 82, 83].

To account for the **deviation of individual** neurons' firing rates from spike-timing variability in spiking network simulations, we also computed the MSE across the entire network (Figure 4.1D, dark green),

$$MSE_{pop} = \frac{1}{N}\sum_{j=1}^{N}(r_j - r_j^0)^2$$

where r_j is the firing rate of neuron $j = 1, \ldots, N$ and r_j^0 is its target rate. Both measures of MSE show a decrease during training and a sharp increase in response to the mismatched stimulus, but MSE_{pop} is larger overall due to the spike-timing variability of each neuron.

4.3.2.1 Mean-field Rate Model Approximation

The results from the spiking network can be understood using a simpler dynamical mean-field model in which mean firing rates of each population are approximated by a system of differential equations,

$$\tau \odot \frac{d\mathbf{r}}{dt} = -\mathbf{r} + f\left(W\mathbf{r} + \mathbf{X}\right) \tag{4.6}$$

where $\tau = [\tau_{e_1} \quad \tau_{e_2} \quad \tau_i]^T$ is a vector of time constants, $\odot$ represents element-wise multiplication, and $\mathbf{r} = [r_{e_1} \quad r_{e_1} \quad r_i]^T$ is a vector approximating the mean firing rates of the two excitatory sub-populations and the inhibitory population. The mean external input to each population is given by the vector

$$\mathbf{X} = \begin{bmatrix} X_{e_1} \\ X_{e_2} \\ X_i \end{bmatrix}$$

and the recurrent connectivity matrix is defined by

$$W = \begin{bmatrix} w_{e_1e_1} & w_{e_1e_2} & w_{e_1i} \\ w_{e_2e_1} & w_{e_2e_2} & w_{e_2i} \\ w_{ie_1} & w_{ie_2} & w_{ii} \end{bmatrix}$$

where [3, 9, 10, 29, 67, 68]

$$w_{ab} = N_b p_{ab} j_{ab}$$

Here, N_b is the number of neurons in population $b = e_1, e_2, i$ (so $N_{e_1} = N_{e_2} = N_e/2 = 2000$ and $N_i = 1000$), p_{ab} is the connection probability from population b to population a, and j_{ab} is the mean non-zero synaptic weight (mean of J_{ab}^{jk} between connected neurons). The inhibitory entries, w_{ai} for $a = e_1, e_2, i$, are negative and evolve according to

$$\frac{dw_{ai}}{dt} = -\eta_a (r_a - r_0^a) r_i \tag{4.7}$$

where η_a sets the timescale of plasticity and r_0^a is the target rate of population $a = e_1, e_2, i$. For simplicity, we consider a rectified linear "f-I" curve,

$$f(I) = \begin{cases} gI & I > 0 \\ 0 & I \leq 0 \end{cases}. \tag{4.8}$$

The gain, g, was fit to spiking network simulations (see Materials and Methods).

Simulating this model shows excellent agreement with the firing rates from the

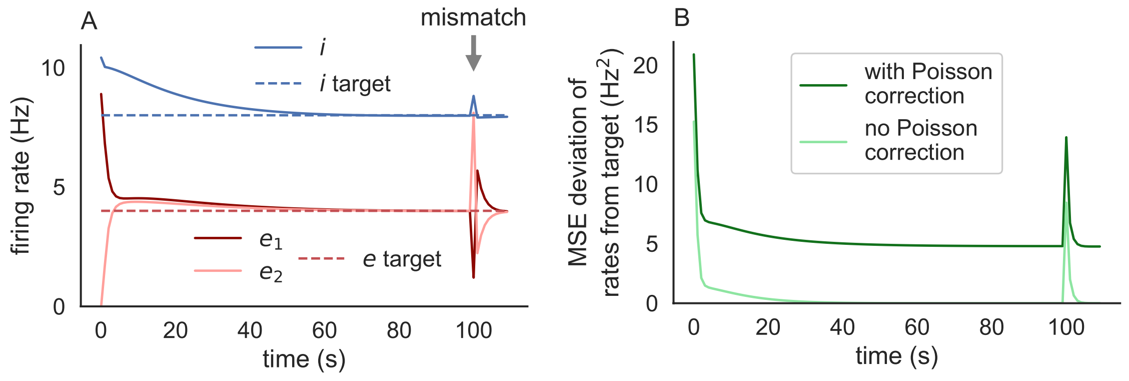

Figure 4.2. **mean-field firing rate model captures the dynamics of the spiking network model.** A: Firing rates of the mean-field firing rate model defined by Eqs. (4.6) and (4.6). Compare to Figure 4.1C. B: MSE deviation of the firing rates from their targets (MSE_{mf}; light green) and the MSE with a Poisson correction ($MSE_{Poisson}$; dark green). Compare to Figure 4.1D.

spiking network simulations (Figure 4.2) and the mean-field simulations are computationally more efficient than the spiking network simulations by a factor of 70 (6.0s for the mean-field simulation compared to 435.0s for the spiking network simulation). The deviation of the firing rates in the mean-field rate model from their targets can be quantified by

$$MSE_{mf} = \sum_{a=e_1,e_2,i} q_a(r_a - r_a^0)^2 \tag{4.9}$$

which is identical to MSE_{mean} above except that r_a represents the rate from the mean-field simulations instead of the mean firing rates from the spiking net simulations. Indeed, MSE_{mf} closely matches MSE_{mean} from the spiking network simulations (Figure 4.2B, compare to Figure 4.1C), demonstrating that the two models have similar mean-field dynamics. The value of MSE_{pop} from the spiking network simulations does not have a direct analogue in the mean-field model, but under an assumption of Poisson-like spike-timing variability in the spiking network, MSE_{pop}

can be approximated by

$$MSE_{Poisson} = MSE_{mf} + \frac{1}{T}\sum_a q_a r_a \tag{4.10}$$

where r_a is the firing rate of population $a = e_1, e_2, i$ from the mean-field model and T is length of the time window over which firing rates are computed in the spiking network simulations. Specifically, $MSE_{Poisson}$ represents the population-level MSE (*i.e.*, MSE_{pop}) that would be produced by populations of Poisson spike trains with firing rates r_a.

Proof. Consider a population of N neurons divided into M sub-populations where sub-population a contains N_a neurons for $a = 1, \ldots, M$ ($M = 3$ and $a = e_1, e_2, i$ for the models considered in this paper). Assume that each neuron in population a spikes like a Poisson process with a rate of r_a. Let n_j^a be the number of spikes emitted by neuron $j = 1, \ldots, N_a$ in population $a = 1, \ldots, M$ during a time interval of duration T and let

$$r_j^a = \frac{n_j^a}{T}$$

be the sample firing rate of neuron j. Then each n_j^a has expectation and variance

$$E[n_j^a] = var(n_j^a) = r_a T$$

be the sample firing rate of neuron j. Then each n_j^a has equal expectation and variance as

$$E[n_j^a] = var(n_j^a) = r_a T$$

so each sample rate has an expectation

$$E[r_j^a] = E\left[\frac{n_j^a}{T}\right] = r_a$$

and variance

$$var(r_j^a) = var\left(\frac{n_j^a}{T}\right) = \frac{r_a}{T}.$$

Now suppose we have target rates of r_a^0 for each neuron in population a and we would like to compute the population-wide MSE deviation of the sample rates from their targets. This can be written as

$$\begin{aligned} MSE_{pop} &= \frac{1}{N}\sum_{j=1}^{N}(r_j - r_j^0)^2 \\ &= \frac{1}{N}\sum_{a=1}^{M}\sum_{j=1}^{N_a}(r_j^a - r_a^0)^2 \\ &= \sum_{a=1}^{M} q_a \frac{1}{N_a}\sum_{j=1}^{N_a}(r_j^a - r_a^0)^2 \end{aligned}$$

where $q_a = N_a/N$ is the proportion of neurons in population a, r_j is the sample rate, and r_j^0 is the rate parameter for neuron $j = 1, \ldots, N$. The inner sum can be written as

$$\begin{aligned} \frac{1}{N_a}\sum_{j=1}^{N_a}(r_j^a - r_a^0)^2 = (r_a^0 - r_a)^2 \\ + \frac{1}{N_a}\sum_{j=1}^{N_a}(r_j^a - r_a)^2 - 2(r_a^0 - r_a)(r_j^a - r_a). \end{aligned}$$

The first term in the sum is the sample variance of r_j^a, so

$$\frac{1}{N_a}\sum_{j=1}^{N_a}(r_j^a - r_a)^2 \approx var(r_j^a) = \frac{r_a}{T}.$$

when N_a is large. The last term in the sum can be ignored when N_a is large because

$$\begin{aligned} \frac{1}{N_a}\sum_{j=1}^{N_a}(r_a^0 - r_a)(r_j^a - r_a) = (r_a - r_a^0)\left(r_a - \frac{1}{N_a}\sum_{j=1}^{N_a} r_j^a\right) \\ \approx 0 \end{aligned}$$

since r_a is the expected value of r_j^a. Putting this all together gives

$$\begin{aligned} MSE_{pop} &\approx \sum_{a=1}^{M} q_a \left[(r_a - r_a^0)^2 + \frac{r_a}{T}\right] \\ &= MSE_{mf} + \frac{1}{T}\sum_{a=1}^{M} q_a r_a \end{aligned}$$

where

$$MSE_{mf} = \sum_{a=1}^{M} q_a (r_a - r_a^0)^2$$

is the mean-field MSE defined in Eq. (4.9). This calculation motivates the definition of the Poisson-corrected MSE,

$$MSE_{Poisson} = MSE_{mf} + \sum_{a=1}^{M} \frac{q_a r_a}{T}$$

as defined in Eq. (4.10). Specifically, our calculations above show that $MSE_{Poisson}$ approximates the population-level MSE (*i.e.*, MSE_{pop}) that would be produced if all of the spike trains in each sub-populations were Poisson processes. The approximation becomes exact as $N_a \to \infty$.

Indeed, $MSE_{Poisson}$ shows close agreement with MSE_{pop} (Figure 4.2B, compare to Figure 4.1D), demonstrating that the deviation of MSE_{pop} away from the values of MSE_{mean} is consistent with Poisson-like spike-timing variability.

This example shows that homeostatic inhibitory synaptic plasticity can train a network to detect mismatched stimuli, which is a form of predictive coding. To better understand how and why the network is able to detect mismatched stimuli, we consider a fixed point analysis via separation of timescales.

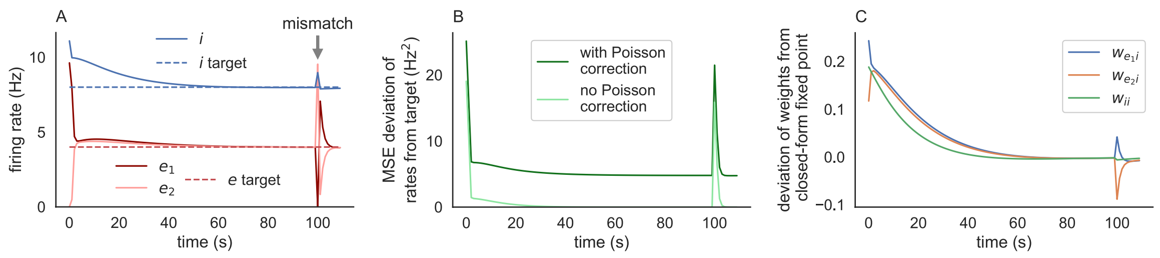

Figure 4.3. **Slow dynamics are captured by a separation-of-timescales approximation.** A: Firing rates of the model defined by Eqs. (4.12). Compare to Figures 4.1C and 4.2A. B: MSE deviation of the firing rates from their targets (MSE_{mf}; light green) and the MSE with a Poisson correction ($MSE_{Poisson}$; dark green) from the model defined by Eqs. (4.12). Compare to Figures 4.1D and 4.2B. C: Deviation of the inhibitory weights, w_{ai}, from the fixed point values given in Eqs. (4.14).

4.3.2.2 Separation of Timescales Approximation

In the absence of plasticity (W fixed, *e.g.*, $\eta_e = \eta_i = 0$), fixed point firing rates would satisfy $\mathbf{r}_0 = f(W\mathbf{r}_0 + \mathbf{X})$. Taking the rectified linear f-I curve from the dynamical mean-field model, if there were a fixed point with positive rates ($r_a > 0$ for all a) then it would be unique and given (as a function of W) by

$$\mathbf{r}(W) = [D - W]^{-1}\,\mathbf{X} = A\mathbf{X} \tag{4.11}$$

where $D = (1/g)Id$ is a diagonal matrix, Id is the identity matrix, and $A = [D - W]^{-1}$. With W fixed, the Jacobian matrix for the firing rate equation, Eq. (4.6), would be given by

$$\mathbf{J} = g\begin{bmatrix} (w_{e_1e_1} - 1)/\tau_e & w_{e_1e_2}/\tau_e & w_{e_1i}/\tau_e \\ w_{e_1e_2}/\tau_e & (w_{e_1e_1} - 1)/\tau_e & w_{e_1i}/\tau_e \\ w_{ie_1}\tau_i & w_{ie_2}\tau_i & (w_{ii} - 1)/\tau_i \end{bmatrix}$$

If the eigenvalues of this matrix have a negative real part, then the fixed point given by Eq. (4.11) is stable and globally attractive.

Due to plasticity, W itself is time-dependent, so this fixed point analysis does not tell the full story. When plasticity is much slower than the firing rate dynamics (η sufficiently small and τ sufficiently large, but η should not be compared directly to τ because they have different dimensions), we can perform a separation of timescales under which $\vec{r}$ relaxes to the quasi-steady-state value given by evaluating Eq. (4.11) at the current value of W, while W evolves more slowly according to Eq. (4.7). Putting this together, the separation of timescales approximation is defined by

$$\begin{aligned} \frac{dW}{dt} &= \begin{bmatrix} 0 & 0 & -\eta_e(r_{e_1} - r_0^e)r_i \\ 0 & 0 & -\eta_e(r_{e_2} - r_0^e)r_i \\ 0 & 0 & -\eta_i(r_i - r_0^i)r_i \end{bmatrix} \\ \mathbf{r} &= \begin{bmatrix} r_{e_1} \\ r_{e_2} \\ r_i \end{bmatrix} = [D - W]^{-1}\,\mathbf{X} = \mathbf{A}\mathbf{X} \end{aligned} \tag{4.12}$$

Note that this is a 3-dimensional dynamical system because $\mathbf{r}$ is defined by a functional relationship instead of differential equations. Solving Eqs. (4.12) directly gives similar results to the full mean-field model and is 482 times more computationally efficient than the full mean-field simulations (Figure 4.3A, B; 12.5×10^{-3}s to simulate Eqs. (4.12) versus 6.0s for the full mean-field model) primarily because the slower dynamics allow for a larger time discretization (we used $dt = 0.1$ms for the full mean-field and $dt = T = 1$s to simulate Eqs. (4.12)). Simulating Eqs. (4.12) was 34751 times faster than the spiking network simulations. This speedup is not surprising given the lower dimension (2 versus 5000 dimensions) as well as the larger time discretization.

During training, $\mathbf{X}$ is fixed to the "matched" value given by Eq. (4.3). During this phase, the slow-timescale system described by Eqs. (4.12) has a fixed point for which $\mathbf{r} = \mathbf{r}^0$ where

$$\mathbf{r^0} = \begin{bmatrix} r_e^0 \\ r_e^0 \\ r_i^0 \end{bmatrix}$$

is a vector of the target rates from the plasticity rule. However, this expression gives the fixed point in terms of $\mathbf{r}$ whereas the dynamical system is described by the dynamics of the entries of W. If the network converges to the target rates during training, then the weight matrix, W, for the slow system converges to a value, W^0 (or, equivalently, A converges to a value of A^0) that satisfies

$$\left[D - W^0\right]^{-1} \mathbf{X}^m = A^0 \mathbf{X}^m = \mathbf{r}^0 \tag{4.13}$$

where

$$\mathbf{X}^m = \begin{bmatrix} X_e^0 + U \\ X_e^0 + V \\ X_i^0 \end{bmatrix}$$

is the value of $\mathbf{X}$ for matched stimuli. Eq. (4.13) is a system of three equations for three unknowns ($w_{e_1 i}$, $w_{e_2 i}$, w_{ii}) and its solution is given by

$$\begin{aligned} w_{e_1 i} &= \frac{r_e^0 - 2g r_e^0 w_{ee} - g(U + X_e^0)}{g r_i^0} \\ w_{e_2 i} &= \frac{r_e^0 - 2g r_e^0 w_{ee} - g(V + X_e^0)}{g r_i^0} \\ w_{ii} &= \frac{r_i^0 - 2 r_e^0 w_{ie} + X_i^0}{g r_i^0} \end{aligned} \tag{4.14}$$

Indeed, the weights converged toward these fixed point values during the training period (before the mismatch stimulus; Figure 4.3C).

When the input is changed by a mismatched stimulus (so $\mathbf{X}$ changes away from its value during training), firing rates deviate from their targets. Using the same quasi-steady state approximation, we can quantify the magnitude of this deviation as

$$\begin{aligned} d\mathbf{r} &:= \mathbf{r}^{mm} - \mathbf{r}^0 \\ &= A^0\mathbf{r}^{mm} - \mathbf{r}^0 \\ &= A^0(\mathbf{X}^{mm} - \mathbf{X}^m) \\ &= A^0 d\mathbf{X} \end{aligned} \tag{4.15}$$

where $\mathbf{r}^{mm}$ is the vector of firing rates during a mismatched trial, $\mathbf{r}^0 = [r_e^0 \;\; r_i^0]^T$ is the vector of target rates, and

$$d\mathbf{X} = \mathbf{X}^{mm} - \mathbf{X}^{\mathbf{m}} = \begin{bmatrix} 0 \\ -V \\ 0 \end{bmatrix}$$

is the perturbation of the external stimulus away from its training value during the mismatched trial. This derivation makes it clear that larger perturbations of the stimulus (larger values of $\|d\vec{X}\|$) generally lead to larger deviations of the firing rates from their targets (larger values of $\|d\vec{r}\|$). Here and elsewhere, $\|\cdot\|$ refers to the Euclidean norm.

Firing rate perturbations, $\|d\mathbf{r}\|$, are especially large if the input perturbations, $d\mathbf{X}$, point in a direction in which $A^0 d\mathbf{X}$ is large. Such directions correspond to the directions indicated by the largest eigenvalue(s) of A^0. Since $A^0 = [D - W^0]^{-1}$, when W^0 is much larger than D in magnitude, these directions correspond to directions indicated by the smallest eigenvalue(s) of W^0. This phenomenon is an instance of "imbalanced amplification" in which a perturbation that points toward the nullspace or "approximate nullspace" of the connectivity matrix, W^0, is amplified by the network, see [29] for more in-depth explanations.

Temporarily ignoring the direction of the perturbation, we can make the rough approximation that $\|d\mathbf{r}\|$ is approximately proportional to $\|d\mathbf{X}\|$. This rough approximation provides the intuition for mismatched responses shown in the simulations above. Put simply, mismatched responses are caused by the deviation of a stimulus away from its “matched” training value, and the magnitude of the mismatched response increases with the magnitude of the input perturbation. While this intuition may seem trivial for this example, its extensions will help explain some non-trivial, counterintuitive results below.

4.3.3 Distributed and Time-constant Inputs

The example above modeled a stimulus that was homogeneous across each neural population, *i.e.*, every neuron in population e_1 received the same input and every neuron in population e_2 received the same input. Stimulus representations in cortical circuits can be distributed in an inhomogeneous way across neural populations [75]. We next considered a spiking network model with distributed bottom-up and top-down inputs (Figure 4.4A). As above, matched and mismatched stimuli were defined by the presence and absence of top-down input to population e_2 (Eqs. (4.3) and (4.5)) to match the bottom-up input to population e_1, but these inputs are heterogeneous vectors ($\vec{U}$ and $\vec{V}$) instead of homogeneous scalars (U and V). Specifically, matched and mismatched stimuli to excitatory neurons were defined by

$$X_e = X_e^0 + \vec{U} + \vec{V} \left.\vphantom{\vec{V}}\right\} \text{ matched} \tag{4.16}$$

and

$$X_e = X_e^0 + \vec{U} \qquad \left.\vphantom{\vec{U}}\right\} \text{ mismatched.} \tag{4.17}$$

where $\vec{U}$ and $\vec{V}$ are normally distributed N_e-dimensional vectors,

$$\begin{aligned} \vec{U} &\sim \sigma_s N(0,1) \\ \vec{V} &\sim \sigma_s N(0,1). \end{aligned} \tag{4.18}$$

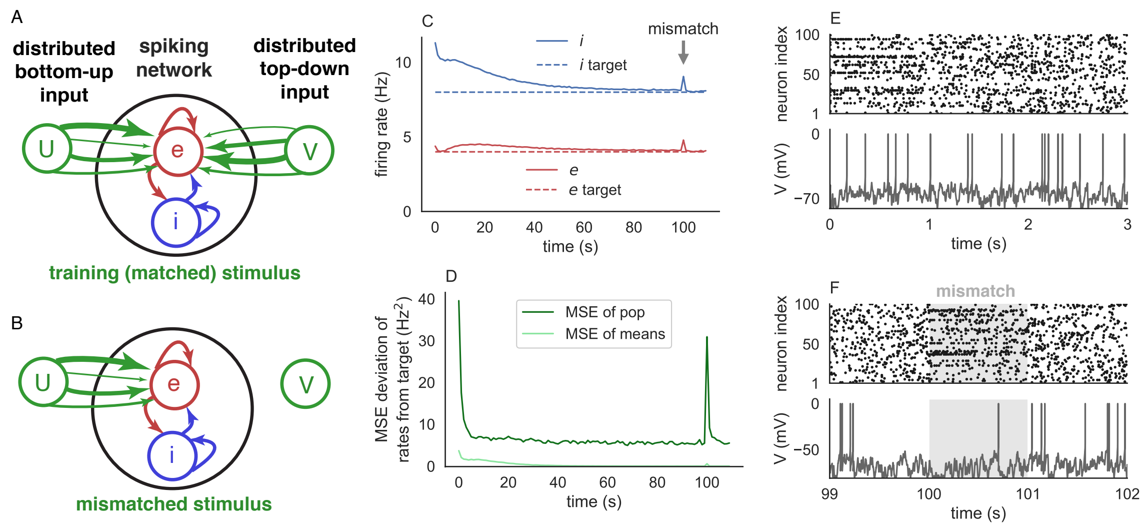

Figure 4.4. **Prediction errors after training on distributed time-constant inputs.** Same as Figure 4.1 except bottom-up and top-down inputs were modeled as distributed stimuli using multivariate Gaussian inputs vectors (Eq. (4.18)).

Here, $N(0,1)$ is a standard multivariate normal distribution and $\sigma_s = X_e^0/5$ controls the strength of the stimuli. Importantly, this means that each neuron receives a different value of top-down and bottom-up input, in contrast to the previous example (Eq. (4.4) and Figures 4.1–4.3) in which every neuron in the same excitatory sub-population received the same input.

Simulating this spiking network model shows that population-averaged firing rates converge to their targets during training on matched stimuli, as expected, but only deviate slightly from their targets in response to a mismatched stimulus (Figure 4.4C).

We suspected that the deviation of mean excitatory and inhibitory firing rates was small because some neurons increased their firing rates and some neurons decreased their firing rates in response to mismatched stimuli, so the increases and decreases canceled at the level of population averages. Another way to see this is to note that the expected value of $\vec{U}$ and $\vec{V}$ is zero, so the absence of $\vec{V}$ does not affect the population-averaged value of the inputs and (under a linear approximation) we should not expect a change in mean firing rates by removing $\vec{V}$. Under this reasoning, the firing rates of individual neurons would still change in response to a mismatched stimulus because individual elements of $\vec{V}$ are non-zero. This line of reasoning implies that MSE_{mean} should not increase much for a mismatched stimulus, but MSE_{pop} should increase more for a mismatched stimulus. Indeed, this is exactly what we observed in simulations (Figure 4.4D).

In summary, our network model with iSTDP learned to adjust inhibitory weights in such a way as to "match" or "cancel" top-down input with bottom-up input in the sense that the firing rates approach their target rates in response to matched stimuli after sufficient training. Moreover, the network responded to mismatched stimuli with deviations in the firing rates away from their target values. Note that the deviation of firing rates from their targets is not a consequence of the mismatch alone, but is due to the network being trained on matched stimuli. In this sense, the network is simply detecting deviations in its input patterns from the input patterns on which it was trained.

4.4 Undetectable Prediction Errors After Training

While instructive, the examples above were restricted to input patterns that were held fixed during training. In other words, the network only learned to associate *one* bottom-up input, U, with *one* top-down input, V (as schematized in Figure 4.1A, B and 4.4A, B). Since animals are exposed to multiple stimuli, a more realistic model would be trained on multiple pairings of top-down and bottom-up inputs. For example, in the visuomotor system, head motion (which we can interpret as top-down input, V) is coupled with the movement of an animal's visual stimulus (which we can interpret as bottom-up input, U). But head motion varies in direction and speed, and the movement of a visual scene covaries with it. Prediction errors arise whenever the learned covariation between head motion and visual stimulus is violated, *i.e.*,

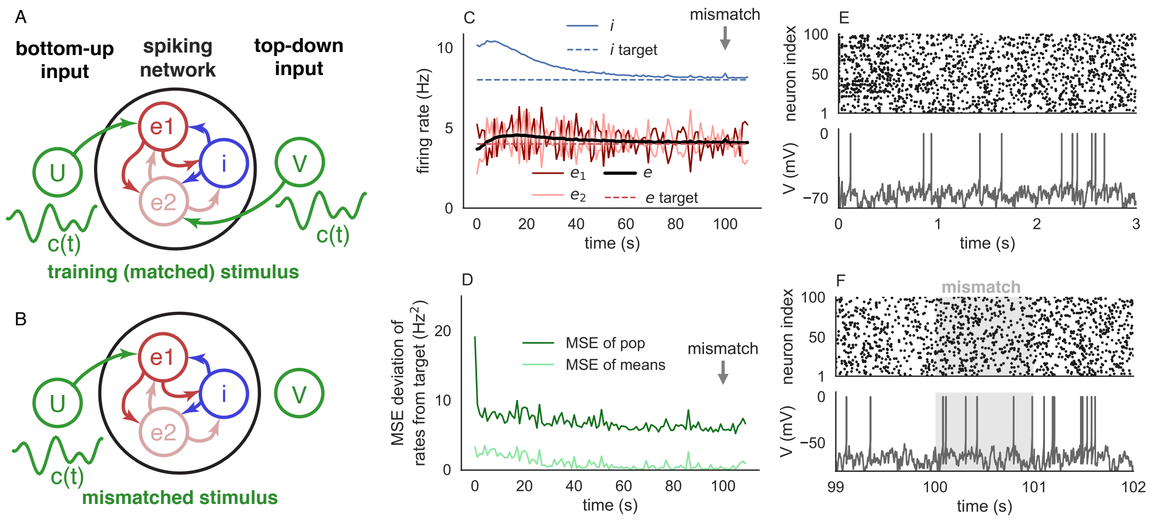

Figure 4.5. **Undetectable prediction errors in a model with time-varying stimuli.** A and B: Network schematic. Same as Figure 4.1A except for the magnitude of the top-down and bottom-up stimuli were multiplied by the same time-varying signal, $c(t)$. C-F: Same as Figure 4.1C-F except we additionally plotted the mean excitatory firing rates (black curve in C).

whenever there is a mismatch between top-down and bottom-up input [8, 45, 47, 53].

4.4.1 Time-varying Inputs

We next considered a simple extension of the first input model from Figures 4.1–4.3 to account for top-down and bottom-up inputs with time-varying intensity. Specifically, the excitatory neurons were again broken into two sub-populations, e_1 and e_2. During training, each neuron in populations e_1 and e_2 received external stimuli of the form (Figure 4.1A)

$$\left.\begin{aligned} X_{e_1} &= X_e^0 + c(t)U \\ X_{e_2} &= X_e^0 + c(t)V \end{aligned}\right\} \text{ matched} \tag{4.19}$$

where $c(t)$ is a scalar time series that changes on each trial. Specifically, $c(t)$ is drawn independently from a uniform distribution on $[0, 2]$ at the start of each 1s trial. Hence, the expected value of $c(t)$ is 1, and therefore, the expected values of X_{e_1} and X_{e_2} are the same as in the example from Figures 4.1–4.3, but they vary around this expectation across time. We used similar top-down and bottom-up, but needed to make the inputs weaker to avoid very large rate deviations,

$$\begin{aligned} U &= X_e^0/20 \\ V &= -X_e^0/20. \end{aligned} \tag{4.20}$$

Hence, bottom-up input, $c(t)U$, is matched by top-down input, $c(t)V$, during training. After training on matched stimuli, we again modeled mismatched stimuli by the absence of top-down input

$$\left.\begin{aligned} X_{e_1} &= X_e^0 + U \\ X_{e_2} &= X_e^0. \end{aligned}\right\} \text{mismatched.} \tag{4.21}$$

The input to e_1 is not out of the ordinary during a mismatched stimulus (it corresponds to the value when $c(t) = 1$ is equal to its expectation) and the input to e_2 is not out of the ordinary either (it corresponds to the value when $c(t) = 0$), the joint value of the inputs to e_1 and e_2 together is out of the ordinary because the inputs are not matched (see Figure 4.5A for a schematic).

We reasoned that if our iSTDP rule could learn the relationship between top-down and bottom-up input during training, then it would detect the mismatch between them by evoking a larger deviation of firing rates from their targets. In other words, the network should detect the out-of-distribution input represented by a mismatch. However, our spiking network simulations contradicted this prediction. Firing rates deviated from the targets even during matched stimuli and the deviation in response to a mismatched stimulus was similar in magnitude (Figure 4.5B–F). Hence, the response to a mismatched stimulus was not detectable in the sense that it could not be distinguished from the response to matched stimuli.

4.4.2 A Mean-field Explanation for the Absence of Detection

We now return to our mean-field theory to better understand why we do not see mismatch responses after training on time-varying inputs, but we do see them after training on time-constant inputs. We first simulated the dynamical rate model from Eqs. (4.6)–(4.8) with the time-dependent stimuli defined by Eqs. (4.19)–(4.21). As above, the dynamical mean-field rate model captured the general trends from the spiking network simulations (compare Figure 4.6A,B to Figure 4.5C,D). Eq. (4.11) for the quasi-steady-state firing rates generalizes to

$$\mathbf{r}(W) = [D - W]^{-1}\,\mathbf{X}(\mathbf{t}) = \mathbf{A}\mathbf{X}(\mathbf{t}) \tag{4.22}$$

4.4.2.1 Timescale Assumptions of the Stimuli

An assumption underlying Eq. (4.22) is that $\mathbf{X}(t)$ changes more slowly than the timescales (τ_a for $a = e, i$) at which firing rates evolve. This assumption is valid in our case because $\mathbf{X}(t)$ switches every 1s while $\tau_a \leq$ 6ms.

Now we can transition to the slower timescale dynamics of W by re-writing Eqs. (4.12) as

$$\begin{gathered}\frac{dW}{dt} = \begin{bmatrix} 0 & 0 & -\eta_e(r_{e_1}(t) - r_0^e)r_i(t) \\ 0 & 0 & -\eta_e(r_{e_2}(t) - r_0^e)r_i(t) \\ 0 & 0 & -\eta_i(r_i(t) - r_0^i)r_i(t) \end{bmatrix} \\ \mathbf{r}(\mathbf{t}) = \begin{bmatrix} r_{e_1}(t) \\ r_{e_2}(t) \\ r_i(t) \end{bmatrix} = [D - W]^{-1}\,\mathbf{X}(t) = A\mathbf{X}(t)\end{gathered} \tag{4.23}$$

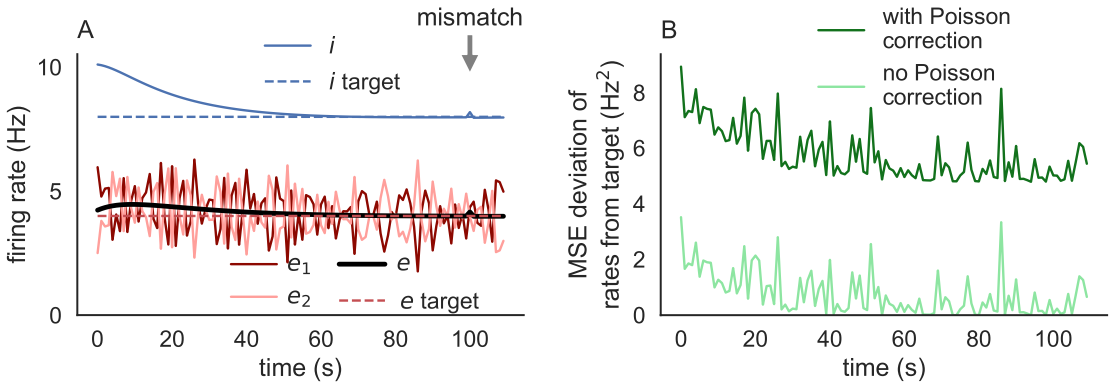

Figure 4.6. **Mean-field rate model with time-varying stimuli.** Same as Figure 4.2 except using the time-varying stimuli from Figure 4.5.

where we have only added the explicit time dependence. Simulating this system shows general agreement with the trends from the spiking networks simulations and the dynamical mean-field model (Figure 4.7A,B, compare to Figure 4.5C,D and Figure 4.6A,B).

4.4.2.2 Separation of Timescales over Mean Approximation

Due to the time-dependence of $\mathbf{X}(t)$ in the current example, Eqs. (4.23) do not have a fixed point, so we cannot proceed directly with the fixed point analysis from above. To perform a fixed point analysis on W, we must assume that plasticity is

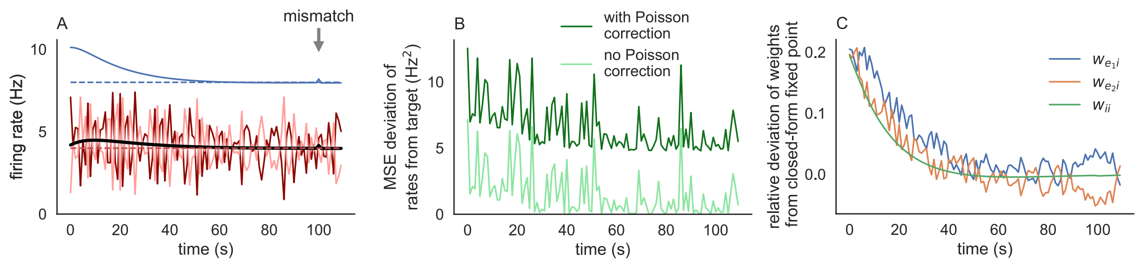

Figure 4.7. **Slow dynamics captured by a separation of timescales in a model with time-dependent stimuli.** Same as Figure 4.3 except using the time-varying stimuli from Figure 4.5A-B.

slower than the stimulus, *i.e.*, that $W(t)$ changes much more slowly than $\mathbf{X}(t)$. This assumption is valid for our simulations and even more so for biological neural circuits. Under this assumption, the slow timescale dynamics of W evolve based on the mean value of $\mathbf{X}(t)$. Specifically, we can use the approximation

$$\begin{aligned} \frac{dW}{dt} &= \begin{bmatrix} 0 & 0 & -\eta_e(\overline{r}_{e_1} - r_0^e)\overline{r_i} \\ 0 & 0 & -\eta_e(\overline{r}_{e_2} - r_0^e)\overline{r_i} \\ 0 & 0 & -\eta_i(\overline{r}_i - r_0^i)\overline{r_i} \end{bmatrix} \\ \overline{\vec{r}} &= \begin{bmatrix} \overline{r}_{e_1} \\ \overline{r}_{e_2} \\ \overline{r}_i \end{bmatrix} = [D - W]^{-1}\overline{\mathbf{X}} = A\overline{\mathbf{X}} \end{aligned} \tag{4.24}$$

where

$$\overline{\mathbf{X}} = E_t[\vec{X}(t)]$$

and E_t denotes the expectation over time during training, *i.e.*, during matched stimuli.

During training (for matched stimuli), we have from Eq. (4.19) that

$$\mathbf{X}^m(t) = \begin{bmatrix} X_e^0 + c(t)U \\ X_e^0 + c(t)V \\ X_i^0 \end{bmatrix} \tag{4.25}$$

Since $E_t[c(t)] = 1$, we have that

$$\overline{\mathbf{X}} = \begin{bmatrix} X_e^0 + U \\ X_e^0 + V \\ X_i^0 \end{bmatrix} \tag{4.26}$$

which is the same as the model from Figures 4.1–4.3. Hence, under this approximation, W should converge to the same fixed point in Eq. (4.14). Notably, this implies that the time-averaged rates should be equal to the target rates, $\overline{\mathbf{r}} = \mathbf{r}^0$. As predicted, simulations show that average firing rates are close to their targets (Figure 4.7A) and the weights do converge to the given fixed point with the addition of some noise (Figure 4.7C) coming from the noisy time-dependence of $\mathbf{X}(t)$ and $\mathbf{r}(t)$.

Therefore, the state of the network (as represented by W) after training is similar for the networks with time-constant and time-dependent stimuli. As a result, the deviation, $d\mathbf{r}(t)$, of the firing rates from their targets on any given trial takes the same form derived in Eq. (4.15),

$$\begin{aligned} d\mathbf{r}(t) &:= \mathbf{r}(t) - \mathbf{r}^0 \\ &= A^0\mathbf{X}(t) - \vec{r}^0 \\ &= A^0(\mathbf{X}(t) - \overline{\mathbf{X}}) \\ &= A^0 d\mathbf{X}(t) \end{aligned} \tag{4.27}$$

where $d\mathbf{X}(t) = \mathbf{X}(t) - \overline{\mathbf{X}}$ is the deviation of the stimulus from the mean value it takes during training and A^0 is the fixed point of $A = [D - W]^{-1}$ after training (see Eq. (4.13) and surrounding discussion). This conclusion assumes that the mean-field approximation in Eq. (4.22) is approximately accurate or, more specifically, that the firing rate response to a perturbation is approximately a linear function of the input perturbation. This, in turn, requires that the input perturbation is not too strong.

As a heuristic, we can ignore the effect of A^0 in Eq. (4.6) and make the approximation that $d\mathbf{r}(t)$ is larger whenever $d\mathbf{X}(t)$ is larger. In other words,

$$\begin{aligned} \|d\mathbf{r}(t)\| &= \|A^0 d\mathbf{X}(t)\| \\ &\approx \|A^0\|\|d\mathbf{X}(t)\| \\ &\propto \|d\mathbf{X}(t)\|. \end{aligned} \tag{4.28}$$

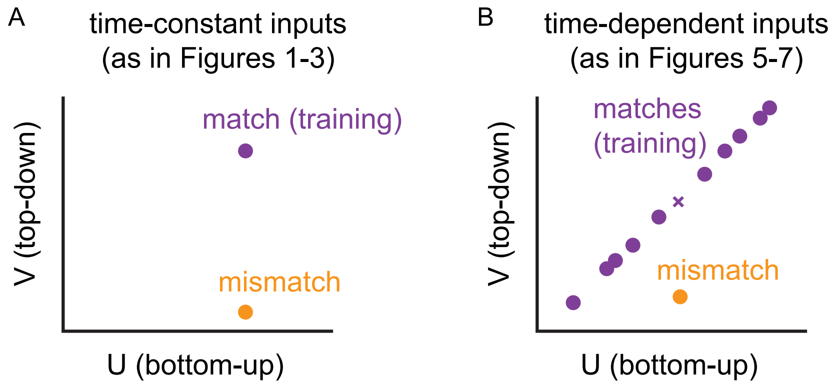

Figure 4.8. **Schematic illustrating why mismatch responses are detectable after training on time-constant, but not time-dependent stimuli.** A: Schematic representing inputs to the network in a model with time-constant stimuli. Training stimuli occupy a single point in (U, V) space (purple dot). The deviation of firing rates from their targets on any particular trial is approximately proportional to the distance of the input from its value during training (Eq. (4.15)). Since the mismatch stimulus (orange dot) is far from the matched, training stimulus, firing rates deviate from their target in response to the mismatched stimulus (as seen in Figures 4.1–4.3). B: Schematic representing inputs to the network in a model with time-varying stimuli. Training stimuli (purple dots) vary in (U, V) space along a predictable line. The mismatched stimulus lies far from this line. However, the deviation of firing rates from their targets on any particular trial is approximately proportional to the distance of the input from its *mean* value during training (Eq. (4.15)). Since the distance between the mismatch input (orange dot) and the mean training stimulus (purple x) is similar to the typical distance between the individual training stimuli (purple dots) and the mean training stimulus (purple x), the deviation of the firing rates from their targets is similar for matched and mismatched stimuli.

where $\|A^0\|$ denotes the induced Euclidean norm on A^0. In other words, stimuli that are further from the mean training stimuli evoke larger firing rates. Note that we necessarily have $\|A^0 d\mathbf{X}(t)\| \leq \|A^0\|\|d\mathbf{X}(t)\|$, so this assumption is saying that $\|A^0 d\mathbf{X}(t)\|$ is not much smaller than $\|A^0\|\|d\mathbf{X}(t)\|$. This approximation assumes that $d\mathbf{X}(t)$ is not close to being orthogonal to the rows of A^0.

During matched stimuli, combining Eqs. (4.25) and (4.26) gives the perturbation for training stimuli

$$d\mathbf{X}^m(t) = \begin{bmatrix} (1-c(t))U \\ (1-c(t))V \\ 0 \end{bmatrix}.$$

Since $|U| = |V|$, we have

$$\begin{aligned} \|d\mathbf{X}^m(t)\|^2 &= 2(1-c(t))^2|V| \\ &= 2u^2(t)|V| \end{aligned}$$

where $u(t) = 1 - c(t)$ is uniformly distributed on $[-1, 1]$. Hence, the squared distance of $\mathbf{X}(t)$ from its mean varies between 0 and $2|V|$. During the mismatched stimulus, we have from Eq. (4.21), that

$$\mathbf{X}^{mm} = \begin{bmatrix} X_e^0 + U \\ X_e^0 \\ X_i^0 \end{bmatrix}$$

Combining this with Eq. (4.26) shows that, during a mismatched stimulus, the input perturbation is

$$d\mathbf{X}^{mm} = \begin{bmatrix} 0 \\ -V \\ 0 \end{bmatrix}$$

and therefore

$$\|d\mathbf{X}^{mm}\|^2 = |V|.$$

Hence, the deviation of the external input, $\mathbf{X}(t)$, from its mean value during training is similar in magnitude during matched and mismatched stimuli. As a result, the deviation of the firing rates from their targets is also similar during matched and mismatched stimuli, so the mismatch is not detectable based on the deviation of firing rates from their targets alone.

This intuition, and how it differs from the time-constant model of Figures 4.1–4.3, is illustrated in Figure 4.8. For the model with time-constant inputs, there is only one stimulus during matched, training trials (Figure 4.8A, purple dot). Since the mismatch stimulus is far from this matched stimulus, the firing rate deviates from its target in response to the mismatched stimulus (as demonstrated in Figures 4.1–4.3). For the model with time-varying stimuli, there are multiple training stimuli that lie along a line (Figure 4.8B, purple dots). While the mismatch stimulus is clearly away from this line (Figure 4.8B, orange dot), the deviation of the firing rates from their targets is approximately proportional to how far an input is from the *mean* training stimulus (Figure 4.8B, purple x). Since this distance is similar for the mismatch stimulus and a typical training stimulus, the deviation of the firing rates from their targets is also similar during matched and mismatched stimuli (as demonstrated in Figures 4.5–4.7). While this intuition might seem obvious in hindsight, the complexity of dynamics in recurrent spiking neural network models can make this conclusion difficult to foresee without the benefit of the mean-field analysis provided here.

4.4.3 Distributed and Time-varying Inputs

For the sake of completeness, we also considered a model with distributed, time-varying stimuli. Specifically, we combined the time-varying stimuli from the example

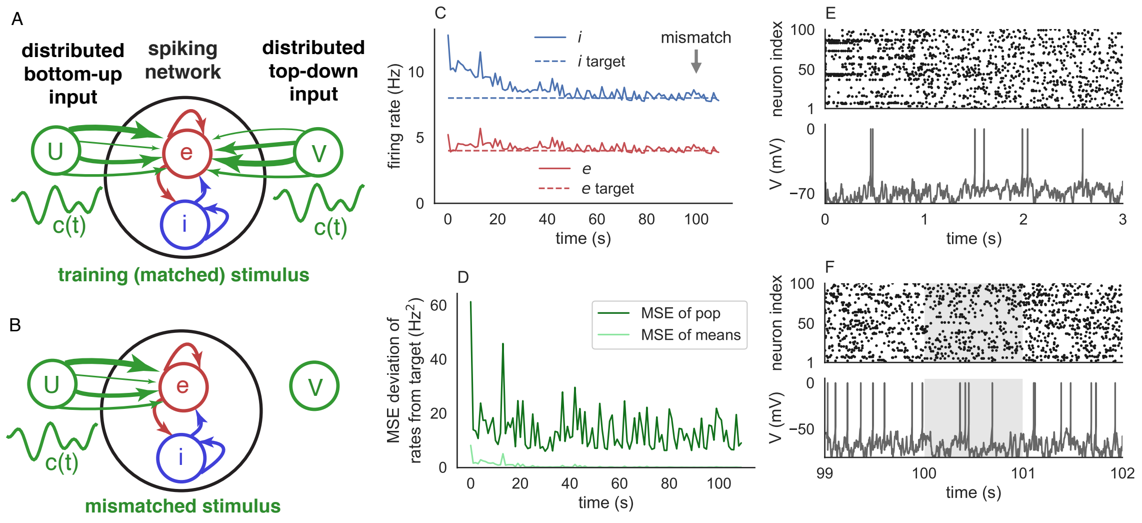

Figure 4.9. **Prediction errors after training on distributed time-dependent inputs.** Same as Figure 4.4 except bottom-up and top-down inputs were time-dependent, as described by Eqs. (4.29)–(4.30).

in Figure 4.7 with the distributed stimuli from the example in Figure 4.4 to get inputs of the form (Figure 4.9A,B)

$$X_e = X_e^0 + c(t)\vec{U} + c(t)\vec{V} \Big\} \text{ matched} \tag{4.29}$$

and

$$X_e = X_e^0 + \vec{U} \quad \Big\} \text{ mismatched.} \tag{4.30}$$

where $c(t)$ is a scalar drawn from a uniform distribution on $[0, 2]$ on each trial, and $\vec{U}$ and $\vec{V}$ are normally distributed N_e-dimensional vectors as in Eq. (4.18). Unsurprisingly, given the failure of the simpler example discussed above, the spiking network model did not produce an easily detectable response to mismatched stimuli (Figure 4.9C-F). Specifically, the deviation of the firing rates away from their targets was similar in matched and mismatched trials (Figure 4.9C,D).

4.5 Generalization

The mean-field analysis above relied on several assumptions that were used to derive approximations. This raises the question of how general our conclusions are. Specifically, for which network models does the argument above imply an absence of noticeable mismatch responses? To answer this question, we can distill the argument above into three fundamental assumptions:

1. The linear approximation in Eq. (4.6) should be approximately accurate,

   $$d\mathbf{r}(t) \approx A^0 d\mathbf{X}(t)$$

   While this assumption is strong, it should be satisfied when $d\mathbf{X}(t)$ is sufficiently small. In addition, balanced excitation and inhibition linearize the firing rate responses of networks to external input [2, 29, 52, 56, 72, 83], so this assumption should hold in networks with balanced excitation and inhibition, which is encouraged by inhibitory synaptic plasticity [3, 10, 40, 85].

2. The approximation in Eq. (4.28) should be accurate, specifically

   $$\|A^0 d\mathbf{X}(t)\| \approx \|A^0\| \|d\mathbf{X}(t)\|$$

   which requires that $d\mathbf{X}(t)$ not be close to orthogonal to the rows of A^0.

3. The magnitude of the input perturbations for a mismatched stimulus should be similar to a typical value during matched stimuli,

   $$\|d\mathbf{X}^{mm}\| \approx \|d\mathbf{X}^{m}(t)\|$$

In general, if a model satisfies these three assumptions then $d\mathbf{r}(t)$ is similar in magnitude during matched and mismatched stimuli. Note that these assumptions are sufficient, but not necessary for a lack of mismatch responses. For example, if assumption 1 is violated because the rate perturbations are nonlinear, then the nonlinear model might still not compute mismatch responses.

Strictly speaking, assumption 2 is stronger than needed. Instead, we only need that the relationship between A^0 and $d\mathbf{X}$ is similar for matched and mismatched stimuli, *i.e.*, that

$$\frac{\|A^0 d\mathbf{X}^m(t)\|}{\|A^0\|\|d\mathbf{X}^m(t)\|} \approx \frac{\|A^0 d\mathbf{X}^{mm}\|}{\|A^0\|\|d\mathbf{X}^{mm}\|}$$

which is a weaker assumption because it allows for $d\mathbf{X}$ to be aligned with the rows of A^0 so long as the alignment is similar for matched and unmatched stimuli.

For our examples in which the network is trained on time-constant input (Figures 4.1–4.4), we have that $\mathbf{X}^m(t) = \overline{\mathbf{X}}$, so $d\mathbf{X}^m(t) = 0$ whereas $d\mathbf{X}^{mm} \neq 0$, so assumption 3 above is not met. This explains why our examples trained on time-constant were able to produce robust mismatch responses.

In previous work [42], a network with homeostatic plasticity successfully computed prediction errors after training on time-varying stimuli. In that work, the weights of the connectivity matrix were carefully chosen so that A was singular and the directions of the input perturbation during matched stimuli (the "feedback" stimulus condition) were in the null space of A^0. See equation 28 in their appendix and note that A^0 was called W in their analysis. As a result, the model studied there does not satisfy assumption 2 above. This explains how [42] were able to compute prediction errors with time-varying inputs.

4.5.1 Networks with External Input to Inhibitory Populations

In all of the examples we have considered so far, external input was provided to excitatory neurons only. However, our analysis implies that our overall results should still hold if the input is provided to inhibitory neurons as well. Specifically, in Eqs. (4.6) and the surrounding equations and analysis, there is nothing preventing $d\mathbf{X}(t)$ from having a non-zero component for the inhibitory population(s). To verify this prediction, we repeated all of the spiking network simulations (those in Figures 4.1, 4.4, 4.5, and 4.9) in models in which external input was also added to the inhibitory population. Our results show the same overall conclusions for all fig-

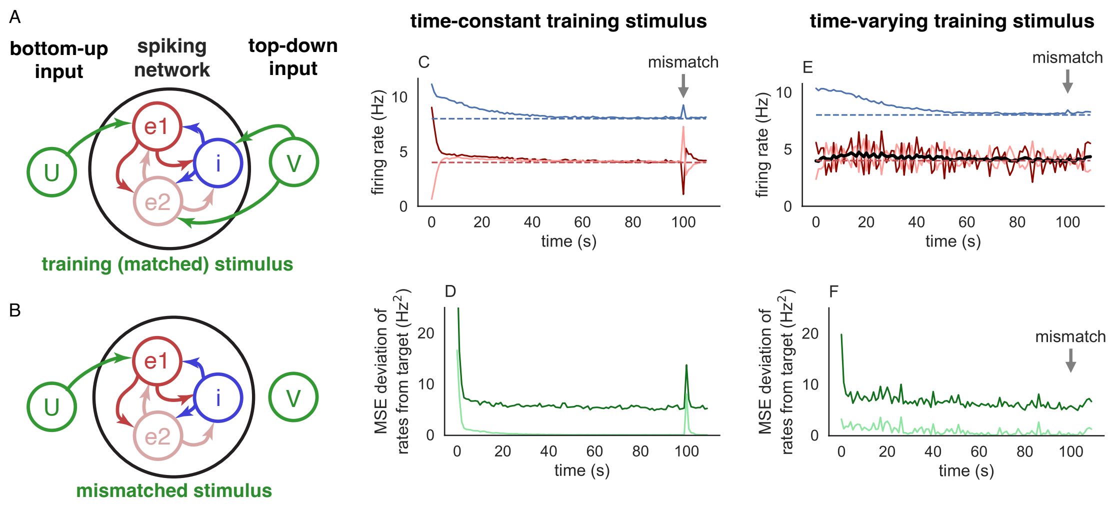

Figure 4.10. **Similar results are obtained with input to inhibitory populations.** A, B: Network schematics. Same as Figure 1A, B except top-down external input was provided to the inhibitory population as well. C, D: Same as Figure 1C, D except top-down external input was provided to the inhibitory population as well. E, F: Same as Figure 4.5C, D except top-down external input was provided to the inhibitory population as well.

ures. Specifically, in all examples, a noticeable mismatch response was observed after training on time-constant inputs, but not after training on time-varying inputs.

We empirically test whether our conclusions were sensitive to the assumption that only excitatory neurons received external input by repeating some simulations from the main text with external input provided to the inhibitory populations as well. Figure 4.10 shows results from a simulation in which top-down input was provided to the inhibitory population in addition to population e_2 during training. For the mismatched stimulus, the top-down input was removed from the inhibitory population and from population e_2. Specifically, in Figure 4.10C,D, we used

$$\left.\begin{aligned} X_{e_1} &= X_e^0 + U \\ X_{e_2} &= X_e^0 + V \\ X_i &= X_i^0 + V \end{aligned}\right\} \text{matched}$$

and

$$\left.\begin{aligned} X_{e_1} &= X_e^0 + U \\ X_{e_2} &= X_e^0 \\ X_i &= X_i^0 \end{aligned}\right\} \text{mismatched}$$

where

$$U = X_e^0/5$$
$$V = -X_e^0/5.$$

which is identical to Figure 4.1 from the main text, but with input V provided to i as well in Figure 4.10E,F, we used

$$\left.\begin{aligned} X_{e_1} &= X_e^0 + c(t)U \\ X_{e_2} &= X_e^0 + c(t)V \\ X_i &= X_i^0 + c(t)V \end{aligned}\right\} \text{matched}$$

and

$$\left.\begin{aligned} X_{e_1} &= X_e^0 + U \\ X_{e_2} &= X_e^0 \\ X_i &= X_i^0 \end{aligned}\right\} \text{mismatched.}$$

where

$$U = X_e^0/20$$
$$V = -X_e^0/20.$$

This is identical to Figure 4.5 from the main text, but with input, V provided to i as well. Our results in Figure 4.10) show a strong mismatch response after training on time-constant input, but not time-varying input.

We additionally tested whether similar results were obtained for distributed external input provided to the inhibitory and excitatory populations in Figure 4.11.

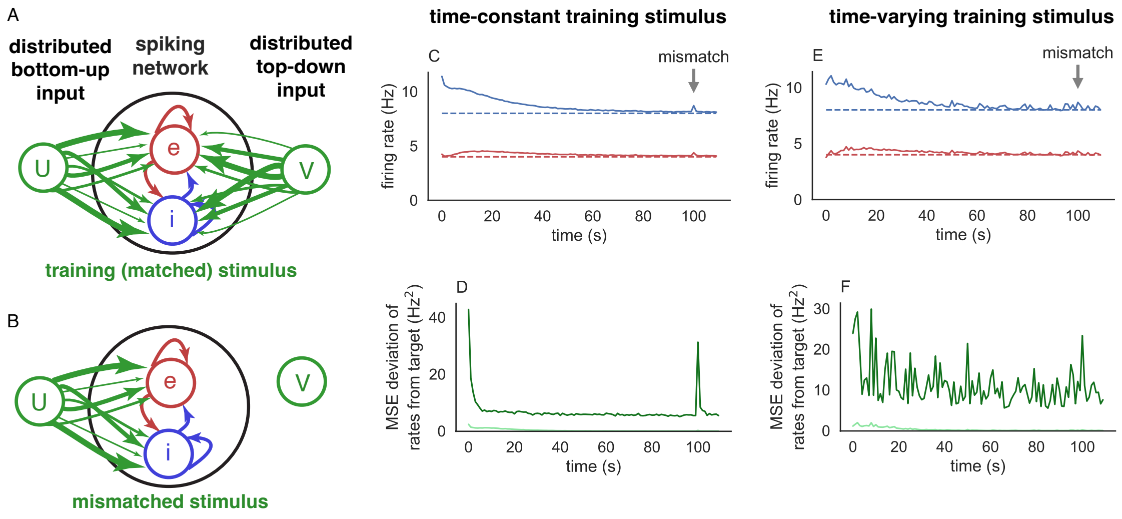

Figure 4.11. **Similar results are obtained with distributed inputs to inhibitory populations.** A, B: Network schematics. Same as Figure 4A, B except distributed external input was provided to the inhibitory population as well. C, D: Same as Figure 4.4C, D except top-down external input was provided to the inhibitory population as well. E, F: Same as Figure 4.9C, D except top-down external input was provided to the inhibitory population as well.

Specifically, in Figure 4.11C,D, we used

$$\left.\begin{aligned} X_e &= X_e^0 + \vec{U}_e + \vec{V}_e \\ X_i &= X_i^0 + \vec{U}_i + \vec{V}_i \end{aligned}\right\} \text{matched}$$

and

$$\left.\begin{aligned} X_e &= X_e^0 + \vec{U}_e \\ X_i &= X_i^0 + \vec{U}_i \end{aligned}\right\} \text{mismatched.}$$

where $\vec{U}_a$ and $\vec{V}_a$ are normally distributed N_a-dimensional vectors,

$$\begin{aligned} \vec{U}_a &\sim \sigma_s N(0,1) \\ \vec{V}_a &\sim \sigma_s N(0,1) \end{aligned}$$

for $a = e, i$. This is identical to Figure 4.4 from the main text except distributed input was provided to the inhibitory population as well. In Figure 4.11E,F, we used

$$\left.\begin{aligned} X_e &= X_e^0 + c(t)\vec{U}_e + c(t)\vec{V}_e \\ X_i &= X_i^0 + c(t)\vec{U}_i + c(t)\vec{V}_i \end{aligned}\right\} \text{matched}$$

and

$$\left.\begin{aligned} X_e &= X_e^0 + \vec{U}_e \\ X_i &= X_i^0 + \vec{U}_i \end{aligned}\right\} \text{mismatched.}$$

which is identical to Figure 9 from the main text except distributed input was provided to the inhibitory population as well. Our results in Figure 4.11 show a strong mismatch response after training on time-constant distributed input, but not time-varying distributed input.

In conclusion, adding external input to the inhibitory population does not qualitatively affect our overall findings.

4.5.2 Networks with Increasing Mismatch Stimuli

Assumption 3 above implies that mismatch responses could be possible after training on time-varying stimuli if the mismatch stimulus is larger in magnitude than the matched stimuli used during training. While this is not necessarily a surprising finding (a larger stimulus should evoke a larger response), we decided to test it in a simulation. Specifically, we repeated the simulation from Figure 4.5, but we scaled the magnitude of the mismatched input by a factor of six. These simulations confirm that a mismatch response was produced in this case.

We next repeated the simulation from Figure 4.5 of the main manuscript but increased the strength of the mismatched stimulus. In particular, we set

$$\left.\begin{aligned} X_{e_1} &= X_e^0 + c(t)U \\ X_{e_2} &= X_e^0 + c(t)V \\ X_i &= X_i^0 \end{aligned}\right\} \text{ matched}$$

and

$$\left.\begin{aligned} X_{e_1} &= X_e^0 + 6U \\ X_{e_2} &= X_e^0 \\ X_i &= X_i^0 \end{aligned}\right\} \text{ mismatched.}$$

In this case, the mismatched stimulus has a larger magnitude than any of the matched stimuli used for training (in addition to the mismatch that occurs). As predicted, we observed a pronounced mismatch response in this case 4.12)

4.5.3 Networks with More Sub-populations

In all of the examples above, we considered only a single inhibitory population and at most two excitatory populations. In reality, there are multiple inhibitory neuron subtypes in the cortex and previous work on mismatch responses with inhibitory

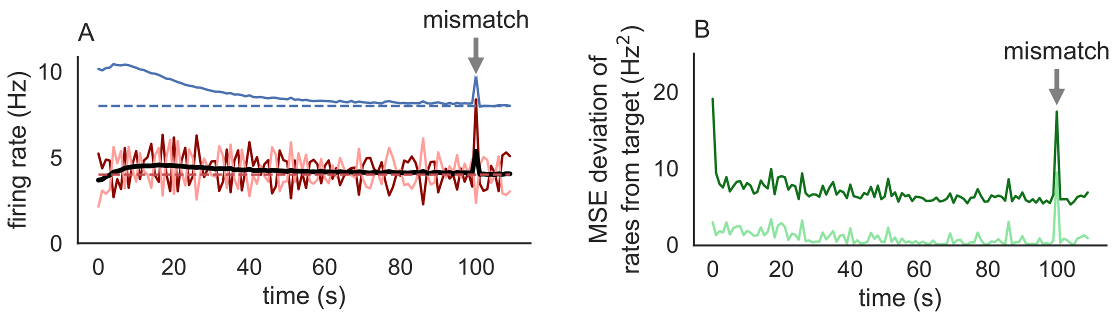

Figure 4.12. **Mismatch responses are observed with stronger mismatched stimuli.** A, B: Same as Figure 4.5 except the strength of the mismatched input was increased by six-fold.

plasticity accounts for this [41, 42]. Our analysis above implies that increasing the number of neuron populations alone should not affect our overall conclusions. To test our findings empirically on a model with several neural populations, we performed a simulation that was identical to the simulation in Figure 4.5 except we used three inhibitory and three excitatory populations. Consistent with our theoretical predictions, the results were qualitatively similar to those in Figure 4.5: After training on time-dependent stimuli, there was no noticeable deviation of firing rates in response to a mismatched stimulus.

In all examples considered so far, we considered a single inhibitory population and one or two excitatory populations. We next tested whether including more populations would affect our results. Specifically, we repeated the simulations from Figure 5, but we broke the excitatory and inhibitory populations each into three subpopulations in Figure 4.13. During training (matched stimuli), populations e_1 and i_1 received bottom-up input from U, and populations e_2 and i_2 received top-down input from V. And populations e_3 and i_3 received no external input. Specifically,

$$\left.\begin{aligned}
X_{e_1} &= X_e^0 + c(t)U \\
X_{e_2} &= X_e^0 + c(t)V \\
X_{e_3} &= X_e^0 \\
X_{i_1} &= X_i^0 + c(t)U \\
X_{i_2} &= X_i^0 + c(t)V \\
X_{i_3} &= X_i^0
\end{aligned}\right\} \text{matched}$$

and

$$\left.\begin{aligned}
X_{e_1} &= X_e^0 + U \\
X_{e_2} &= X_e^0 \\
X_{e_3} &= X_e^0 \\
X_{i_1} &= X_i^0 + U \\
X_{i_2} &= X_i^0 \\
X_{i_3} &= X_i^0.
\end{aligned}\right\} \text{mismatched.}$$

Our results in Figure 4.13C and D show no visible mismatch response, consistent with our original findings from Figure 4.5. Hence, simply adding more populations does not change our overall findings. This is consistent with the conclusions reached by our theoretical arguments in the previous chapters.

4.6 Discussion

We combined numerical simulations of spiking networks and mean-field rate models with mathematical analysis to evaluate the extent to which homeostatic inhibitory synaptic plasticity can train an unstructured network to compute prediction errors. We found that the networks successfully learn to compute prediction errors when

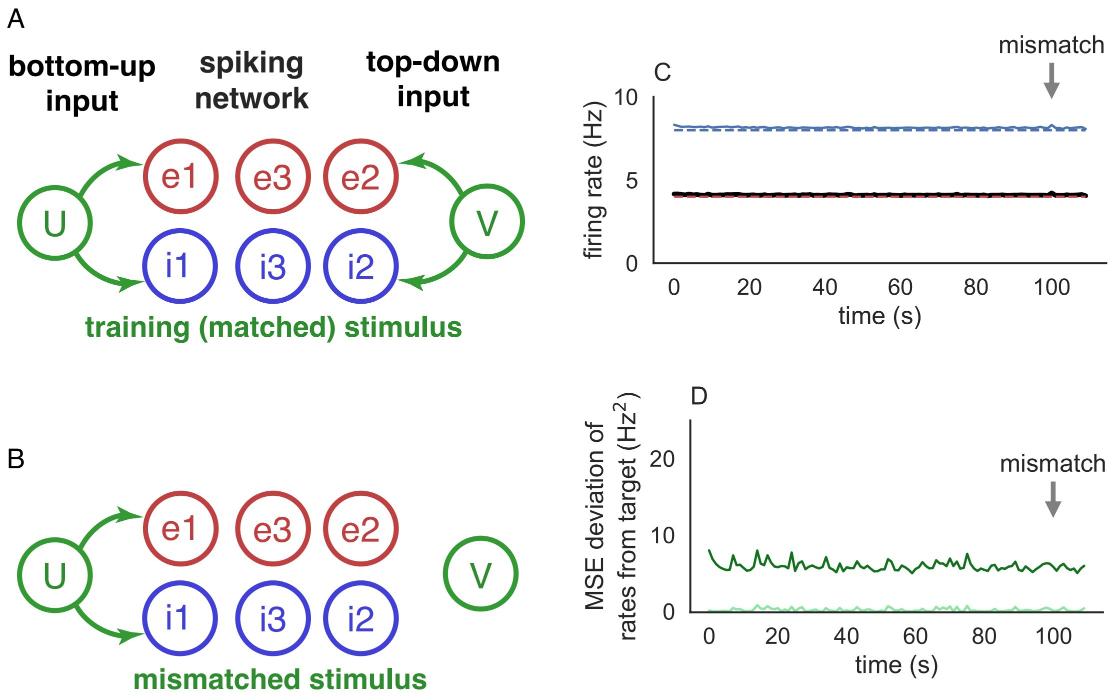

Figure 4.13. **Mismatch responses are not observed when more populations are considered** Same as Figure 4.5 except more populations were added. Connections between populations are not shown for the simplicity of the diagram.

training stimuli are static. Specifically, if top-down and bottom-up inputs are fixed in time during training, then firing rates in the trained network will maintain baseline firing rates in response to stimuli that match the training stimuli, but firing rates will deviate from their baseline levels in response to mismatched stimuli. This result holds when stimuli are uniform (with each of a few sub-populations receiving homogeneous external input) or when stimuli are distributed (with each neuron receiving distinct, but time-constant levels of external input during training).

To our surprise, simulations showed that even under a simple model of time-varying stimuli, in which bottom-up and top-down inputs are modulated by the same time-varying factor, the same networks fail to produce reliable mismatch responses after training. Specifically, firing rates deviate from their baseline levels by a similar amount in response to stimuli that are matched (a shared modulation, as in training) or mismatched (one input is modulated differently than the other). We used a mean-field approximation to explain these empirical findings and elucidate a set of conditions under which robust mismatch responses do not occur. Our results, therefore, help to clarify the extent to which homeostatic inhibitory synaptic plasticity is sufficient to train a network to compute mismatch responses.

For networks trained on time-varying inputs, our results show a lack of mismatch responses in the sense that firing rates do not deviate from their baseline (when a deviation is measured by mean-squared error) more during mismatched inputs than they do for matched stimuli. However, mismatch responses could potentially be detected by some linear projection of the firing rates and this linear projection could be fed as input to a readout neuron that would be able to detect mismatch responses. However, our main goal was to understand the situations under which a natural homeostatic plasticity rule would spontaneously produce elevated responses to mismatched stimuli. Training a separate linear projection is outside the scope of this goal.

Inhibitory homeostatic synaptic plasticity is only one of many homeostatic mechanisms in the brain [81]. While homeostatic plasticity is one candidate mechanism for predictive coding, other homeostatic mechanisms could play a role as well. Future work should consider the potential role of other homeostatic mechanisms in predictive coding and mismatch detection.

Previous work [41, 42] found that networks with homeostatic plasticity *can* learn to compute mismatch responses in models with time-varying stimuli that are similar to the time-varying stimuli that we used (in the cases where our networks failed). They used a more biologically detailed network model with multiple inhibitory subtypes and multi-compartment excitatory neurons. Importantly, connectivity in their model was constrained so that matched stimuli were in the nullspace of the effective connectivity matrix (A in our work, W in theirs). Our theoretical analysis agrees with their analysis showing that this assumption is necessary for their overall results. We additionally provided a set of conditions under which more general classes of models will not produce robust mismatch responses, which generalizes some of the theoretical results in [42] to more general classes of networks. The requirement that matched stimuli are in the null space of the effective connectivity matrix is a strong assumption because it implies that the connectivity matrices must be precisely tuned. Moreover, the dimension of the null space of the connectivity matrix must match the dimensionality of the training stimuli, which could make it difficult to train a network to maintain baseline firing rates on a higher dimensional space of training stimuli.

Our study and the previous work described above [41, 42] incorporates homeostatic synaptic plasticity, but do not account for any other of the wide variety of synaptic plasticity rules observed in neural recordings. Other work has shown that predictive coding can be learned in carefully constructed networks using learning rules that are not exclusively homeostatic [14]. Indeed, our approach of learning prediction errors in unstructured, randomly connected networks could potentially be made successful if the target rates, r_0^a, were effectively modulated by the top-down or bottom-up input. Future work should consider the possibility of learning prediction errors in unstructured, random networks by combining these approaches.

CHAPTER 5

LEARNING FIXED POINTS IN RECURRENT NEURAL NETWORK MODELS

This chapter is adapted from my most recent and ongoing work.

In the previous chapter, we evaluated whether synaptic plasticity can learn to detect some prediction errors in unstructured networks. To continue the same direction of neuron communication and learning, this chapter focuses on learning the fixed point of firing rates through the connection between BRNNs and ARNNs in machine learning that we developed in Chapter 3.

In ARNNs, neurons communicate through activations, *i.e.*, $\mathbf{y} = f(W\mathbf{x})$ where f is a continuous real-valued function such as ReLu, $\mathbf{x}$ is a vector of presynaptic activations, W is a connectivity weight matrix, and $\mathbf{y}$ is the output responses. **Theorem 2**, Eq. (3.5) from our previous work established such a direct relationship between fixed point firing rates in biologically realistic cortical circuit models and fixed point activations in ARNNs [10]. Specifically, hidden state activations in ARNNs with ReLu activations obey equations of the form

$$\mathbf{h}_{t+1} = [W^{rec}\mathbf{h}_t + \mathbf{x}_t]^+ \tag{5.1}$$

We now try to understand how to train the recurrent connections, W, from a static input perspective, $\mathbf{x}_t = \mathbf{x}$, and perform supervised learning on the fixed points, $\mathbf{r}$, with a connectivity weight update in a form of

$$\Delta W = \eta F(W, \mathbf{h}_l, \mathbf{x}).$$

We found that the standard gradient descent with respect to W is ineffective because of singularities that arise in the fixed point equation $\mathbf{y} = f(W\mathbf{y} + \mathbf{x})$. However, gradient descent on a re-parameterized model gives a better update rule for entries of W and more stable learning dynamics. Under a linear approximation, this geometry also performs faster learning. In addition, we have extended the results to properly account for nonlinearities and applied them to non-trivial machine learning benchmarks. These results improve our understanding of synaptic plasticity rules in the brain, and also inspire new learning rules for artificial recurrent neural networks used in machine learning.

5.1 Introduction

RNNs are widely used in machine learning and in computational neuroscience. In machine learning, they are typically used to learn dynamical responses to time series inputs. In computational neuroscience, they are sometimes used to model dynamical responses of neurons to dynamical stimuli [79, 80], but are also often used to model stationary, fixed point neural responses to static inputs. For example, models of visual cortical responses and related phenomena like surround suppression often rely on models of recurrently connected model neurons [10, 24, 29, 30, 65, 74]. Typically, in this approach, the neural response is modeled as a fixed point of the recurrent dynamics.

The natural approach to learning fixed points of recurrent neural networks is to use direct gradient descent on the recurrent weight matrix after the network has converged toward a fixed point. A direct application of this approach, called "truncated backpropagation through time" can be computationally expensive because it requires the application of backpropagation through time on a computational graph unrolled over many time steps. Some numerical approximations have been developed and other approaches have been developed that use the implicit equation for fixed points to derive the exact gradients of the loss with respect to the weight matrix at the exact fixed point, or some approximations to this quantity [4, 55, 64, 66, 92]. These approaches can also be computationally expensive because the gradient derived from the implicit equation involves the matrix inverses, which either need to be computed directly or approximated using, for example, iterative methods.

In this work, we show that, in addition to being computationally expensive, gradient descent on the recurrent weight matrix can lead to poor learning performance because the associated loss landscape has singularities and other features that make it poorly conditioned for gradient-based learning.

Several authors have argued that the default practice of performing gradient descent with respect to the Euclidean gradient of the loss with respect to weights should not always be assumed to be the optimal approach to learning. In machine learning applications, non-Euclidean gradients informed by information theory, such as the natural gradient have been argued to be superior [5, 6, 59]. In computational neuroscience, the use of a Euclidean gradient implicitly assumes a specific choice of units for each quantity in a mode, and, more generally assumes a specific parameterization of the biological model [78]. Different units or different parameterizations of a biological model will yield different gradients and ultimately different learning dynamics. Hence, gradient descent using the Euclidean gradient of the loss with respect to synaptic weights under a specific choice of parameterization and units might not capture learning dynamics or learned representations in biological neuronal networks [78].

We derive learning rules for fixed point of recurrent neural networks under a natural re-parameterization of the network model. The two resulting learning rules can be interpreted as steepest descent and gradient descent on the weight matrix with respect to a non-Euclidean metric and gradient, respectively. We demonstrate empirically that both of these learning rules exhibit more robust learning dynamics than direct gradient descent on the recurrent weight matrix under the Euclidean geometry on weights. Moreover, one of the two new learning rules is computationally more efficient than gradient descent on the recurrent weight matrix under Euclidean geometry because its exact implementation does not require the computation of inverses, the unrolling of a computational graph, or other iterative methods to compute the weight update after fixed points and their loss have been computed.

Our results demonstrate that standard gradient-based learning methods are not ideal for training fixed points of recurrent neuronal networks, but an alternative learning rule is more robust and computationally more efficient.

5.2 Fixed Point Firing Rate Model and Machine Learning Tasks

5.2.1 Firing Rate Model Descriptions

We consider a recurrent neural network (RNN) model of the form [27, 32, 79, 80]

$$\tau \frac{d\mathbf{r}}{dt} = -\mathbf{r} + f(W\mathbf{r} + \mathbf{x}) \tag{5.2}$$

where $\mathbf{r}(t) \in \mathcal{R}^N$ is a vector of model firing rates, $\tau > 0$ is a time constant, $W \in \mathcal{R}^{N\times N}$ is a recurrent connectivity matrix, $\mathbf{x} \in \mathcal{R}^N$ models external input to the network, and $f : \mathcal{R} \to \mathcal{R}$ is a non-negative, non-decreasing activation function or "f-I curve", which is applied point-wise. For a time-constant input, $\mathbf{x}(t) = \mathbf{x}$ in Figure5.1A, the fixed point firing rates satisfy

$$\mathbf{r} = f(W\mathbf{r} + \mathbf{x}). \tag{5.3}$$

5.2.2 Fixed Point Stability

The stability of fixed point firing rates from Eq. (5.2) is determined by the eigenvalues of the Jacobian matrix,

$$J = \frac{1}{\tau}[-I + GW] \tag{5.4}$$

where $G = \mathrm{diag}(f'(\mathbf{z}))$ is a diagonal matrix with entries

$$G_{jj} = f'(\mathbf{z}_j)$$

and $\mathbf{z} = [W\mathbf{r} + \mathbf{x}]$ is the vector of neural inputs or pre-activations evaluated at their fixed points. Specifically, a fixed point is hyperbolically stable if all eigenvalues of J have a negative real part. For a simple example in Figure 5.1B, G is the identity, then the corresponding eigenvalues of J satisfied

$$\Lambda(J) = \frac{1}{\tau}[-1 + \Lambda(W)]$$

where $\Lambda(W)$ denotes the eigenvalues set of W. Hence, stable fixed points require that all the eigenvalues of W have a real part less than 1 (points within the blue unit circle in Figure 5.1B).

5.2.3 Supervised Learning

In machine learning applications, RNNs are typically used to learn mappings from input time series, $\mathbf{x}(t)$, to output time series, $\mathbf{r}(t)$, and they are often trained by using backpropagation through time. In computational neuroscience, RNNs of the form in Eq. (5.2) are often studied for their fixed point properties, for example, to study orientation selectivity and surround suppression among other phenomena [10, 24, 29, 30, 65, 74], but the weights in these studies are often chosen by hand, not learned.

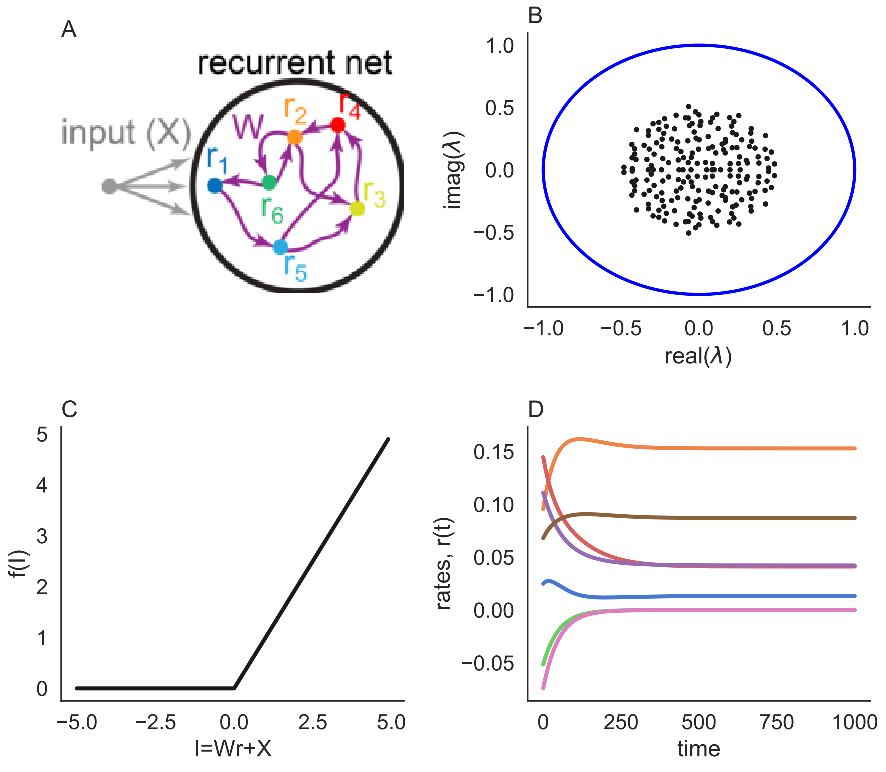

Figure 5.1. **Training fixed point from a recurrent neural network** **A**: Diagram of a recurrent neural network (RNN) with static input, *i.e.,* regression *or* image data trained to produce fixed point firing rate output. **B)**: Eigenvalues of connectivity matrix, W. The unit circle is shown in blue. Stability requires that the eigenvalues of W have a real part less than 1. **C**: Rectified linear function used as an "f-I" curve, where $G = Id$ in Eq.2.22. **D**: Fixed point firing rates using rectified linear simulation for 1000 ms in **C**. The fixed point dynamic described 6 randomly selected neurons of the RNN in **A** with a size of $N = 200$, neurons connectivity are under the stability in **B**, the external input is generated from $N-$dimensional linear regression synthetic data with some noise.

As a combination of these perspectives, we are interested in learning mappings from static inputs, $\mathbf{x}(t) = \mathbf{x}$, to their associated fixed points, $\mathbf{r}$, given by Eq. (5.3) in Figure 5.1A, C and D.

Specifically, consider a supervised learning task with a cost function of the form

$$J = \frac{1}{m}\sum_{i=1}^{m} L(\mathbf{r}^i, \mathbf{y}^i)$$

where $\mathbf{x}^i$ is an input, $\mathbf{y}^i$ is a label, L is a loss function, and $\mathbf{r}^i = f(W\mathbf{r}^i + \mathbf{x}^i)$ is the fixed point that the network converges to under input $\mathbf{x}^i$. This learning task presents unique challenges because fixed points are defined *implicitly* by Eq. (5.3) instead of explicitly as a function of $\mathbf{x}$, and also because we only wish to learn *stable* fixed points.

The data set $\{(\mathbf{x}^i, \mathbf{y}^i)\}_i$ can be the entire data set in the case of full-batch learning or a mini-batch in the case of stochastic learning. Updates to W during learning can be written as

$$W \leftarrow W + \Delta W$$

where

$$\Delta W = \frac{1}{m}\sum_{i=1}^{m} \Delta W^i$$

Here, ΔW^i is an update rule that can depend on $\mathbf{x}^i$, $\mathbf{y}^i$, and $\mathbf{r}^i$. Below, we derive and compare three different update rules, ΔW_1^i, ΔW_2^i, and ΔW_3^i, for minimizing J.

5.2.4 Gradient Descent on the Recurrent Weight.

The first learning rule we consider is the direct gradient descent of the loss surface with respect to W,

$$\Delta W_1^i = -\eta_W \nabla_W L(\mathbf{r}^i, \mathbf{y}^i) \tag{5.5}$$

where $\eta_W > 0$ is a learning rate. If the fixed point, $\mathbf{r}^i$, is hyperbolically stable, then the Jacobian J from Eq. (2.22) has eigenvalues with negative real part, so $I - G^iW = -\tau J$ is invertible and we have (see Appendix A.1)

$$\Delta W_1^i = -\eta_W G^i \left[I - G^iW\right]^{-T} (\nabla_{\mathbf{r}^i} L) \left(\mathbf{r}^i\right)^T. \tag{5.6}$$

where $G^i = \mathrm{diag}(f'(\mathbf{z}^i))$ evaluated at the fixed point $\mathbf{r}^i$ and U^{-T} denotes the inverse transpose of a matrix, U. If $G^i_{jj} \neq 0$ for all j, then $[G^i]^{-1}$ exists so we can simplify Eq. (5.6) to

$$\Delta W_1^i = -\eta_W \left[\left[G^i\right]^{-1} - W\right]^{-T} (\nabla_{\mathbf{r}^i} L) \left(\mathbf{r}^i\right)^T \tag{5.7}$$

Evaluating Eqs. (5.6) and (5.7) is computationally expensive because they require the calculation of a matrix inverse. While some approaches have been developed to make the computation of the update more efficient [4, 55, 64, 66, 92], we also find in examples below that this update rule can lead to poor learning performance. We next propose an alternative learning rule based on a nonlinear reparameterization of the model.

5.3 Newly Developed Learning Rules

5.3.1 Nonlinear Reparameterizing of the RNNs

To motivate the reparameterized model, first consider the special case of a linear network defined by

$$f(x) = x$$

In this case, $G = I$ is the identity matrix and Eq. (5.3) for the fixed point can be written as

$$\mathbf{r} = [I - W]^{-1}\mathbf{x}.$$

This is a linear model in the sense that **r** is a linear function of **x**, but the nonlinear dependence of **r** on W (especially a nonlinearity involving a matrix inverse) makes for the complicated and computationally expensive update from Eq. (5.6). Instead of performing gradient descent with respect to W, we propose instead to first apply a nonlinear change of coordinates to obtain new parameters,

$$A = F(W) := [I - W]^{-1}. \tag{5.8}$$

If we parameterize the model in terms of A instead of W, then fixed points satisfy the standard linear model

$$\mathbf{r} = A\mathbf{x}.$$

Gradient descent of the loss with respect to A gives the standard update rule for a linear, single-layer perceptron

$$\begin{aligned} \Delta A^i &= -\eta_A \nabla_A L(\mathbf{r}^i, \mathbf{y}^i) \\ &= -\eta_A \left(\nabla_{\mathbf{r}^i} L\right) \left(\mathbf{x}^i\right)^T \\ &= -\eta_A \left(\nabla_{\mathbf{r}^i} L\right) \left(\mathbf{r}^i\right)^T A^{-T} \end{aligned} \tag{5.9}$$

where we distinguish between the learning rate, η_A, used for the re-parameterized model, and the learning rate η_W used for the original parameterization. Eq. (5.9) gives a gradient-based update to the new parameter, A, but our original RNN model is parameterized by W. To update our original parameters, we need to change the ΔA from Eq. (5.9) back to W coordinates. To do this, note that we want to find a value for ΔW that satisfies $A + \Delta A = F(W + \Delta W)$ whenever $A = F(W)$ and ΔA comes from Eq. (5.9). In other words, the update to W is given by

$$\begin{aligned} \Delta W_2^i &= F^{-1}(F(W) + \Delta A^i)) - W \\ &= -\left[[I - W]^{-1} - \eta_A \left(\nabla_{\mathbf{r}^i} L\right) \left(\mathbf{r}^i\right)^T [I - W]^T\right]^{-1} + I - W \end{aligned} \tag{5.10}$$

where $F^{-1}(A) = I - A^{-1}$ is the inverse of $F(W)$.

To summarize this approach, if Eq. (5.10) is used to update parameters, W, under the linear fixed point model, $\mathbf{r} = f(W\mathbf{r} + \mathbf{x})$ with $f(x) = x$, then the learning dynamics will be identical to a standard linear regression of parameters, A, on the model $\mathbf{r} = A\mathbf{x}$ using learning rate η_A

Since gradient descent with respect to A in Eq. (5.9) represents steepest descent of the loss surface in the new parameter space of A and since Eq. (5.10) gives the same updates in the original parameter space of W, the learning rule in Eq. (5.5) corresponds to steepest descent of the loss surface using a non-Euclidean metric defined by

$$d(W_1, W_2) = \|F(W_1) - F(W_2)\|_E$$

where $\|B\|_E = \sqrt{\mathrm{Tr}(BB^T)}$ is the Euclidean or Frobenius norm on matrices. Note that $d(\cdot,\cdot)$ is a metric when restricted to the space of all matrices, W, for which $I - W$ is invertible. Hence, if we restrict to W that yield hyperbolically stable fixed points, Eq. (5.5) corresponds to steepest descent with respect to a non-Euclidan metric, but the metric d is not necessarily generated by an inner product, so Eq. (5.10) cannot be called gradient descent on a non-Euclidean *geometry* since the notion of geometry requires a metric induced by an inner product. In Section 5.3.1, we show that an approximation to ΔW_2^i produces gradient descent on non-Euclidean geometry. Moreover, in Section 5.4, we present examples showing that steepest descent with respect to a non-Euclidean metric, as defined by ΔW_2^i, is better suited to learning fixed points than the standard approach to gradient descent represented by ΔW_1^i.

Eq. (5.10) was derived for the specific case $f(x) = x$, but we can extend it to a model with arbitrary $f(x)$. To do so, we first linearize Eq. (5.3) to obtain a linearized fixed point equation,

$$\mathbf{r} = G[W\mathbf{r} + \mathbf{x}] \tag{5.11}$$

which has a closed-form solution given by

$$\mathbf{r} = [I - GW]^{-1} G\mathbf{x}. \tag{5.12}$$

Note, again, that $I - GW$ is invertible whenever $\mathbf{r}$ is a hyperbolically stable fixed point.

Given Eq. (5.12), a natural choice of new parameters would be

$$A = [I - GW]^{-1} G, \tag{5.13}$$

because it would produce a (linearized) model of the form $\mathbf{r} = A\mathbf{x}$. Note that under the linear model $f(z) = z$, we have $G = I$, and recover the parameterization in Eq. (5.8), so Eq. (5.13) is a generalization of Eq. (5.8). However, the update rule to W derived from gradient descent on A from the parameterization in Eq. (5.13) is susceptible to blow up or singularities when some values of $G_{jj} = f'(\mathbf{z}_j)$ become small in magnitude or zero. To see why this is the case, suppose $G_{jj} = \mathcal{O}(\epsilon)$ is small for some j and consider an update to W of the form $W = W + \Delta W$. Then the resulting update to $\mathbf{r}_j$ is, to linear order in ϵ,

$$\begin{aligned} \Delta \mathbf{r}_j &= \sum_k G_{jj} \Delta W_{jk} r_k \\ &= \mathcal{O}(\epsilon \Delta W). \end{aligned}$$

On the other hand, an update of the form $A = A + \Delta A$ gives

$$\begin{aligned} \Delta \mathbf{r}_j &= \sum_k \Delta A_{jk} \mathbf{r}_k \\ &= \mathcal{O}(\Delta A). \end{aligned}$$

Hence, if we want ΔW to produce the same change, $\Delta \mathbf{r}$, produced by ΔA, then we must have $\Delta W \sim \mathcal{O}(\Delta A/\epsilon)$. These large updates to W will cause dramatic changes to W in response to inputs for which G has small elements at the fixed point, ultimately undercutting the model's performance (see Appendix A.2 for more details). In the extreme case that $G_{jj} = 0$ for some j, updates to W do not impact $\mathbf{r}$ (*i.e.*, $\Delta \mathbf{r}_j = 0$ for any ΔW under the linear approximation $\mathbf{r} = G[W\mathbf{r} + \mathbf{x}]$), so we cannot derive a ΔW to match a given ΔA, *i.e.*, the reparameterization in Eq. (5.13) is ill-posed.

To circumvent these problems, we instead take the parameterization

$$A = F(W) := [G - GWG]^{-1} \tag{5.14}$$

in place of Eq. (5.13). Under the linearized fixed point equation in Eq. (5.11), we then obtain the linear model

$$\mathbf{r} = GAG\mathbf{x}. \tag{5.15}$$

This equation is linear in $\mathbf{x}$ and in the new parameters, A. Hence, learning A is again a linear regression problem, albeit with the extra G terms. These extra G terms prevent singularities and blowup when G_{jj} terms become small or zero because $\Delta r_j = \mathcal{O}(\epsilon \Delta A)$ is small whenever we make an update of the form $A = A + \Delta A$ with $G_{jj} = \mathcal{O}(\epsilon)$ small. Under the simple linear model $f(z) = z$, we have $G = I$ and recover the parameterization in Eq. (5.8), so Eq. (5.14) is a generalization of Eq. (5.8) (just like Eq. (5.13) is).

Note that each input (*i.e.*, each i) will potentially have a different gain matrix, $G^i = \text{diag}(f'(\mathbf{z^i}))$, so each sample will have a potentially different value of $A^i = [G^i - G^i W G^i]^{-1}$ as well. The gradient-based update of the loss, $L(\mathbf{r}^i, \mathbf{y}^i)$, with respect to A^i for each sample becomes

$$\begin{aligned}\Delta A^i &= -\eta_A \nabla_{A^i} L(\mathbf{r}^i, \mathbf{y}^i) \\ &= -\eta_A G^i \left(\nabla_{\mathbf{r}^i} L\right) (\mathbf{r}^i)^T [G^i]^{-1} A^{-T}\end{aligned} \tag{5.16}$$

Using the same approach used to derive Eq. (5.10) above, we can again derive an update to W given by

$$\begin{aligned}\Delta W_2^i &= F^{-1}(F(W) + \Delta A^i) - W \\ &= -\left[\left[I - G^i W\right]^{-1} G^i - \eta_A \left[G^i\right]^2 \left(\nabla_{\mathbf{r}^i} L\right) (\mathbf{r}^i)^T \left[I - G^i W\right]^T G^i\right]^{-1} \\ &\quad + \left[[G^i]^{-1} - W\right]\end{aligned} \tag{5.17}$$

This update can only be evaluated directly in the situation where $G^i_{jj} \neq 0$ for all j so that the inverse of the gain matrix, G, exists. However, note that $[W_2^i]_{jk} \to 0$ as $G^i_{jj} \to 0$, as expected, so in situations where $G^i_{jj} = 0$, it is consistent to take $[W_2^i]_{jk} = 0$. Note also that Eq. (5.17) is equivalent to Eq. (5.10) whenever $G = I$, as expected.

5.3.2 Linear Approximation of the Reparameterized Rule

The reparameterized rule in Eq. (5.17) is rather a complicated learning rule, and the matrix inverses can be computationally inefficient. If we assume that $\eta_A > 0$ is small, then we can approximate Eq. (5.17) by applying Taylor expansion to a linear order in η_A. This gives the linearized parameterized rule (see Appendix A.3 for details),

$$\Delta W_3^i = -\eta_A \left[I - W G^i\right] G^i \left(\nabla_{\mathbf{r}^i} L\right) (\mathbf{r}^i)^T \left[I - G^i W\right]^T \left[I - G^i W\right] \tag{5.18}$$

The radius of convergence of the associated Taylor polynomial is given by

$$\left\| \eta_C \left[\left[A^i\right]^{-1} \right] \left[G^i\right]^2 \left(\Delta A^i\right) \left[G^i\right]^2 \right\| < 1. \tag{5.19}$$

Therefore, if

$$\eta_A \ll \frac{1}{\| \left[G^i - G^i W G^i\right] [G^i]^2 \left(\nabla_{\mathbf{r}^i} L\right) \left(\mathbf{r}^i\right)^T \left[G^i - G^i W^T G^i\right] [G^i]^2 \|},$$

the ΔW_3^i from Eq. (5.18) approximates ΔW_2^i from Eq. (5.17).

In contrast to Eqs. (5.6) and (5.17) for ΔW_1^i and ΔW_2^i, Eq. (5.18) for ΔW_3^i does not require the computation of matrix inverses. Like ΔW_1^i and ΔW_2^i, ΔW_3^i satisfies $\Delta W_{jk} \to 0$ whenever $G_{jj} \to 0$, but unlike Eq. (5.17) for ΔW_2^i, Eq. (5.18) for ΔW_3^i can be evaluated directly when $G_{jj} = 0$ for some j.

Recall that ΔW_2^i can be interpreted as steepest descent with respect to a non-Euclidean metric, but it cannot be interpreted as gradient descent because the corresponding metric does not generate an inner product, which is necessary for defining a non-Euclidean gradient.

In contrast, ΔW_3^i can be interpreted as gradient descent of the loss function with respect to W on non-Euclidean geometry. To see why this is the case, first, note that ΔW_3^i is related to ΔW_1^i according to

$$\Delta W_3^i = B^i \Delta W_1^i C^i, \tag{5.20}$$

where

$$B^i = [I - WG^i][I - WG^i]^T$$

and

$$C^i = [I - G^i W]^T [I - G^i W].$$

Here, we took $\eta_A = \eta_W = \eta$ to highlight the relationship between the two update rules, but constant scalar coefficients do not affect these results.

Using Eq. (5.20), we may conclude that ΔW_3^i is equivalent to gradient descent of the loss with respect to W on non-Euclidean geometry. To explain this statement in more detail, note that the gradient of $L(\mathbf{r}^i, \mathbf{y}^i)$ with respect to W depends on the choice of metric or geometry [78]. Given an inner product, $\langle \cdot, \cdot \rangle_a$, on $\mathcal{R}^{N\times N}$ the gradient of a function, $F : \mathcal{R}^{N\times N} \to \mathcal{R}$, on the geometry imposed by $\langle \cdot, \cdot \rangle_a$ evaluated at $W \in \mathcal{R}^{N\times N}$ is the unique matrix $\nabla_W^a F \in \mathcal{R}^{N\times N}$ such that for every $U \in \mathcal{R}^{N\times N}$ [77, 78],

$$\langle \nabla_W^a F, U \rangle_a = \lim_{\epsilon \to 0} \frac{F(W + \epsilon U) - F(W)}{\epsilon}.$$

The standard Euclidean gradient, $\nabla = \nabla^E$, is given by taking the geometry produced by the Euclidean inner product,

$$\langle U, V \rangle_E = \sum_{jk} U_{jk} V_{jk} = \mathrm{Tr}(UV^T).$$

Recall that ΔW_1^i is defined by the Euclidean gradient,

$$\Delta W_1^i = -\eta \nabla_W^E L(\mathbf{r}^i, \mathbf{y}^i)$$

where $L(\mathbf{r}^\mathbf{i}, \mathbf{y}^i)$ is interpreted as a function of W. We claim that

$$\Delta W_3^i = -\eta \nabla_W^B L(\mathbf{r}^i, \mathbf{y}^i) \tag{5.21}$$

where ∇_W^B is the gradient under the geometry defined by the inner product,

$$\langle U, V \rangle_B = \mathrm{Tr}(B^{-1} U C^{-1} V^T) = \langle B^{-1} U, V C^{-1} \rangle_E.$$

Here and below, for notational convenience, we do not write the explicit dependence of B or C on i but note that B and C do depend on i through G^i. In other words, there are distinct matrices, B and C, and therefore distinct inner products, $\langle \cdot, \cdot \rangle_B$ at each gradient descent iteration. Given Eq. (5.20), we can prove Eq. (5.21) by showing that

$$\nabla_W^B L = B \left[\nabla_W^E L \right] C. \tag{5.22}$$

To show Eq. (5.22), first define the $N \times N$ standard basis matrices E^{jk} entry-wise by

$$E^{jk}_{j'k'} = \begin{cases} 1 & j = j' \text{ and } k = k' \\ 0 & \text{otherwise} \end{cases} .$$

for $j, k = 1, \ldots, N$. Now compute the inner product of the gradient with E^{jk},

$$\begin{aligned} \left\langle \left[\nabla^B L \right], E^{jk} \right\rangle_B &= \left\langle B^{-1} \left[\nabla^B L \right], E^{jk} C^{-1} \right\rangle_E \\ &= \operatorname{Tr} \left(B^{-1} \left[\nabla^B L \right] C^{-1} \left[E^{jk} \right]^T \right) \\ &= \sum_{n=1}^{N} \left[B^{-1} \left[\nabla^B L \right] C^{-1} E^{kj} \right]_{n,n} \\ &= \sum_{n,m=1}^{N} \left[B^{-1} \left[\nabla^B L \right] C^{-1} \right]_{n,m} [E_{k,j}]_{m,n} \\ &= \left[B^{-1} \left[\nabla^B L \right] C^{-1} \right]_{jk} \end{aligned} \tag{5.23}$$

where the last line follows from the fact that $E^{kj}_{n,m} = 1$ when $n = k$ and $m = j$, and it is equal to zero for all other j, k. But we also have, from the definition of a gradient,

$$\left\langle \left[\nabla^B L \right], E^{jk} \right\rangle_B = \lim_{\epsilon \to 0} \frac{J(W + \epsilon E^{jk}) - J(W)}{\epsilon} = \frac{\partial J}{\partial W_{jk}} = \left[\nabla^E L \right]_{jk}. \tag{5.24}$$

Since Eqs. (5.23) and (5.24) apply for all indices j, k, we may conclude that

$$B^{-1}\left[\nabla^B L\right]C^{-1} = \left[\nabla^E L\right]$$

and therefore,

$$\left[\nabla^B L\right] = B\left[\nabla^E L\right]C$$

which concludes our proof.

In summary, if W is updated according to ΔW_3^i from Eq. (5.18), then this is equivalent to performing gradient descent on the loss with respect to the weight matrix under the geometry defined by the new inner product, $\langle U, V\rangle_{B^i}$. Below, we present examples showing that this geometry is better suited to learning W than gradient descent with respect to the standard Euclidean geometry. Specifically, ΔW_3^i learns more robustly than ΔW_1^i.

5.3.3 Regularization for Fixed Point Problems

The learning rules derived above were derived explicitly to minimize the cost function without regard for whether the learned fixed points are stable. Regularization can be used to encourage stable fixed points, and to improve generalization.

Specifically, we can modify any of the learned rules above by subtracting a scalar multiple of the weight matrix,

$$\Delta W \leftarrow \Delta W - \lambda W, \tag{5.25}$$

where $\lambda > 0$ is a hyperparameter. The subtraction of λW on each iteration is a form of weight decay [34], which pushes the eigenvalues of W toward zero. This tends to encourage negative eigenvalues of the Jacobian matrix, $J = [-I + GW]/\tau$ since it encourages the $-I$ term to dominate. Therefore, regularization through Eq. (5.25) tends to promote stability when learning fixed points in RNNs.

Notably, ΔW_2^i above was derived to be equivalent to gradient-based learning on a linear model $\mathbf{r} = GAG\mathbf{x}$ (or $\mathbf{r} = A\mathbf{x}$ when $G = I$). When solving this linear problem directly with parameters, A, one would normally apply weight decay to A itself by modifying the gradient-based learning rule according to

$$\Delta A \leftarrow \Delta A - \lambda A. \tag{5.26}$$

Following the approach in Section 5.3.1, we could derive a ΔW that produces the same learning dynamics as this *regularized* version of learning parameters, A. However, this would promote unstable fixed points. To see why this is the case, consider the special case $G = I$, so the $A = [I - W]^{-1}$ and $J = [-I + W]/\tau$. Weight decay on A from Eq. (5.26) pushes the eigenvalues of A toward zero. Since $J = -A^{-1}/\tau$, this would increase the spectral radius of J, promoting eigenvalues with a positive real part and hence unstable fixed points. Hence, we should apply weight decay directly to W, not to A, if we want to learn stable fixed points. As a result, even though ΔW_2^i is equivalent to gradient-based learning of A in the absence of regularization, they are not equivalent when regularization is used.

5.4 Experiments and Results

We next evaluate and interpret each of the learning rules derived above on several supervised learning tasks on increasing complexity.

5.4.1 Example 1: Linear Least Squares Problem in One-dimension.

For illustrative purposes, we first consider a simple linear example in one dimension. In particular, we take $N = 1$,

$$f(z) = z,$$

and

$$L(r, y) = \|r - y\|_2^2 = (r - y)^2. \tag{5.27}$$

We write r and y in regular font instead of boldface because they are scalars. Note that $G = 1$ and $W = w$ are also scalars for this model. The fixed point solution in Eq. (5.3) is given by

$$r = \frac{x}{1 - w}, \tag{5.28}$$

and it is stable if $w < 1$.

To test the gradient-based learning of fixed points in this example, we generated $m = 500$ normally distributed inputs, x_i, with a noisy, linear relationship to the labels, y_i (Figure 5.2A). Specifically,

$$\begin{aligned} x_i &\sim \sigma_x N(0, 1) \\ y_i &\sim \frac{1}{1 - \hat{w}} x_i + \sigma_y N(0, 1) \end{aligned} \tag{5.29}$$

where the $N(0, 1)$ are i.i.d., standard normally distributed random numbers and $\hat{w}$ is the ground truth weight parameter. We used $\hat{w} = -1$, $\sigma_x = 0.1$, and $\sigma_y = 0.01$.

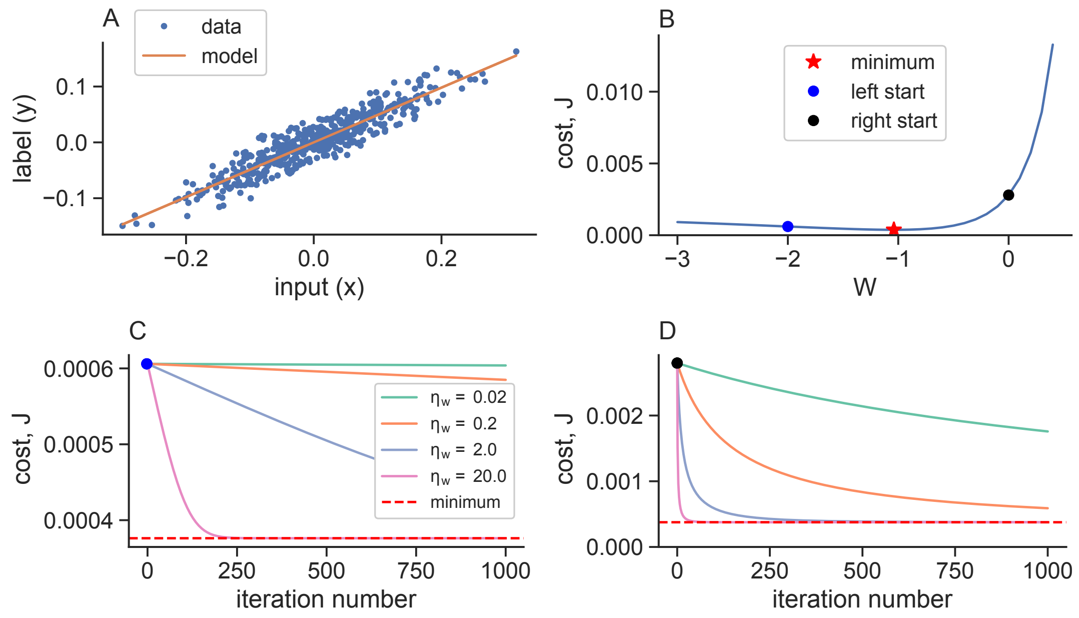

Figure 5.2. **Gradient-based learning of a fixed point firing rate in a one-dimensional linear regression task. A)** A simple linear regression task to find the best-fit line (orange) from a set of 500 randomly generated data points (blue). Model output, r (orange line), is defined by solutions to the fixed point equation $r = wr + x$ where x is the input and w is the model parameter. **B)** The cost as a function of w. The cost is convex and has a minimum value at w^* (red star), but the slope of the landscape is very different on each side of the minimum (*i.e.* $w_{left} = -2$, a blue dot on the left side *vs.* $w_{right} = 0$, black dot on the right side). **C,D)** Gradient descent on w starting from initial values w_{left} and w_{right}, respectively with various learning rates.

5.4.1.1 Learning Through the Gradient Descent Approach

The cost function can be written in matrix form as

$$\begin{aligned} J(w) &= \frac{1}{m}\sum_{i=1}^{m} L(r^i, y^i) \\ &= \frac{1}{m}(R-Y)(R-Y)^T \end{aligned}$$

where $X = [x_1 \;\; x_2 \ldots x_m]$ is the $1 \times m$ array of all inputs (the "design matrix" [61]), Y is the $1 \times m$ array of labels, and $R = X/(1-w)$ is the array of outputs. The gradient-based update from Eq. (5.7) can then be written as

$$\begin{aligned} \Delta w_1 &= \frac{1}{m}\sum_{i=1}^{m} \Delta w_1^i \\ &= \frac{1}{m}\sum_{i=1}^{m} \frac{-2\eta_w}{1-w}(r^i - y^i)r^i \\ &= \frac{-2\eta_w}{m(1-w)}(R-Y)R^T \\ &= \frac{-2\eta_w}{m(1-w)}\left(\frac{X}{1-w} - Y\right)R^T \end{aligned} \tag{5.30}$$

where we used $R = X/(1-w)$ to obtain the last line. This update defines direct gradient descent on the cost function, $J(w)$, with respect to w.

The cost landscape, $J(w)$, is convex on the region $w \in (-\infty, 1]$, over which fixed points are stable (Figure 5.2B; see Appendix A.4 for proof) and the cost function is therefore minimized by solving $\Delta w_1 = 0$ for w to get the minimizer (Figure 5.2B, red star)

$$w^* = 1 - \frac{XX^T}{YX^T}. \tag{5.31}$$

The loss landscape has a vertical asymptote at $w = 1$ (which also specifies the boundary of the stability region) and the slope of the loss landscape is very different on each side of its minimum (Figure 5.2B, blue curve). To demonstrate how this uneven landscape affects gradient-based learning, we performed gradient descent using Δw_1

from Eq. (5.30) with initial conditions on each side of the minimizer: $w_{left,0} = -2$ (Figure 5.2C, corresponding to the blue starting point in Figure 5.2B) and $w_{right,0} = 0$ (Figure 5.2D, corresponding to the black starting point in Figure 5.2B). Due to the differences in steepness on either side of the minimizer, the loss curve decreases at very different speeds when using different initial conditions with the same learning rate. This suggests that, under gradient-based learning of fixed points, a good choice of learning rate can depend sensitively on the choice of initial weights. While this is not necessarily problematic for this simple one-dimensional example, the demonstration of this phenomenon in one dimension will help us understand problems with gradient-based learning of fixed points in higher dimensions that we encounter below.

5.4.1.2 Learning Under the Reparameterized Update.

In one dimension with $G = 1$, the reparameterization in Eqs. (5.8) and (5.14) can be written as (Figure 5.3A)

$$a = F(w) := \frac{1}{1-w}.$$

Under this new parameterization, the fixed point equation in Eq. (5.28) becomes a linear model,

$$r = ax. \tag{5.32}$$

The cost, J, is a quadratic function of the new parameter, a, (Figure 5.3B)

$$J(a) = \frac{1}{m}(aX - Y)(aX - Y)^T.$$

Correspondingly, we have a standard linear model for which the gradient-based update to a is obtained from Eq. (5.9) with $X = R/a$, and this update is indeed the

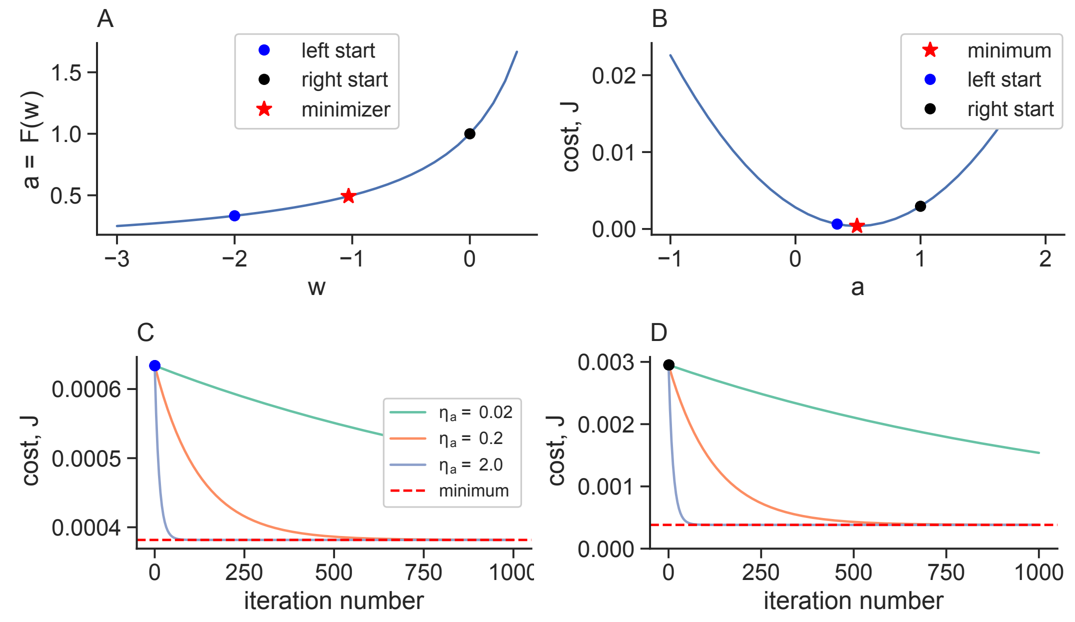

Figure 5.3. **Learning a fixed point firing rates in a one-dimensional linear regression task under a learning rule derived from a re-parameterized model. A)** The new parameter $a = F(w)$ as a function of w. **B)** The cost as a function of a is quadratic. The red star shows the minimum and the blue and black circles show the same initial states, as in Figure 5.2A, mapped to the new parameter space. **C,D)** Same as Figure 5.2C,D except using the learning rule from Eq. (5.35). This is equivalent to using gradient descent on a (*i.e.*, Eq. (5.33)) on the cost landscape in panel A.

delta rule [34, 91],

$$\Delta a = \frac{2}{m}(aX - Y)X^T. \tag{5.33}$$

The cost function is a convex quadratic (Figure 5.3B) and gradient descent with a sufficiently small learning rate is guaranteed to converge to the unique minimzer [13, 49], also see Appendix A.6

$$a^* = F(w^*) = \frac{YX^T}{XX^T}. \tag{5.34}$$

The corresponding update to w, from Eqs. (5.10) and (5.17), is

$$\begin{aligned}\Delta w_2 &= \frac{1}{m}\sum_{i=1}^{m} \Delta w_2^i \\ &= \frac{1}{m}\sum_{i=1}^{m}\left(\frac{-1}{\frac{1}{1-w} - 2\eta_A(r^i - y^i)r^i(1-w)} + 1 - w\right) \\ &= \frac{1}{m}\sum_{i=1}^{m}(1-w)\left(1 - \frac{1}{1 - 2\eta_a\left(x^i x^i - x^i y^i + w x^i y^i\right)}\right)\end{aligned} \tag{5.35}$$

We applied the learning rule in Eq. (5.33) to the same learning task above using initial conditions $a_{left,0} = F(w_{left,0}) = 1/3$ and $a_{right,0} = F(w_{right,0}) = 1$ corresponding to the initial conditions for w used above (Figure 5.3C,D). Since the cost landscape is quadratic, the learning rule in Eq. (5.35) learns at a similar rate for both initial conditions. This suggests that reparameterizing the equation for the fixed point into a linear model can improve learning robustness. We later test this conclusion in a more complex setting by transitioning to a higher-dimensional example.

5.4.1.3 Learning Via Linearizing the Reparameterized Rule

In one dimension, the update ΔW_3 from Eq. (5.18) is given by linearizing the rule from Eq. (5.35) around $\eta_A = 0$ to get

$$\begin{aligned}
\Delta w_3 &= \frac{\eta_a}{\eta_w}(1-w)^2 \Delta w_1 (1-w)^2 \\
&= \frac{-2\eta_a(1-w)^3}{m}(R-Y)R^T \\
&= \frac{-2\eta_a(1-w)^3}{m}(a^2 XX^T - aYX^T) \\
&= \frac{-2\eta_a(1-w)^3}{m}\sum_{i=1}^{m}(a^2 x^i x^i - a y^i x^i)
\end{aligned} \tag{5.36}$$

One interpretation of this update rule is that it represents the gradient-based update, Δw_1, multiplied by a factor, $(1-w)^4$, which depends on w. Note that $(1-w)^4$ is an increasing function of the distance of w from the singularity at $w = 1$. When w is further from $w = 1$, the update Δw_3 is increased more relative to Δw_1. Looking back at Figure 5.2B, this corresponds to taking larger steps whenever the slope of $J(w)$ is small, thereby at least partially correcting the problem of differing slopes on either side of the minimum.

Using Δw_3 from Eq. (5.36) to update w for this one-dimensional linear problem gives similar results to using Δw_2 from Eq. (5.35) (Figure 5.4A, B), as expected. Hence, this linearized re-parameterized update rule also overcomes the problems with Δw_1 from Eq. (5.30) for direct gradient descent on $J(w)$ with respect to the Euclidean gradient on w.

5.4.2 Example 2: Linear Least Squares Problem in Higher-dimensions.

We next generalize the results from the previous section to higher dimensions ($N > 1$). We first consider the case in which there are more neurons than data samples ($N > m$), which we refer to as the "over-parameterized" case.

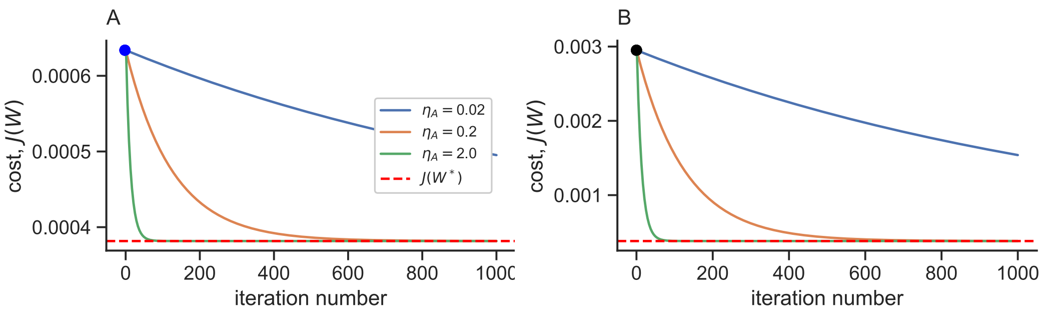

Figure 5.4. **Learning a fixed point firing rates in a one-dimensional linear regression task under a learning rule derived from the linearized re-parameterized model. A, B)** Same as Figure 5.3C, D except using the learning rule from Eq. (5.36).

We used the same cost function in Eq. (5.27), where now the rate and response vector becomes a matrix instead, $R, Y \in \mathcal{R}^{N\times m}$. For $i = 1, \ldots, m$, $\mathbf{y}^i \in \mathcal{R}^N$ is the ith target. We again use $f(z) = z$ so that $G = I$ and fixed point firing rates are given by $\mathbf{r} = [I - W]^{-1}\mathbf{x}$.

For the simulation, we first generate inputs, $X = [\mathbf{x}^1 \;\; \mathbf{x}^2 \;\; \ldots \mathbf{x}^m] \in \mathcal{R}^{N\times m}$, from a Gaussian distribution. Similarly, we generate targets $Y = [\mathbf{y}^1 \;\; \mathbf{y}^2 \;\; \ldots \mathbf{y}^m] \in \mathcal{R}^{N\times m}$ by multiplying inputs by the ground truth matrix, $\hat{W}$, and adding noise. Specifically, we define

$$\begin{aligned} X &\sim \sigma_x N_{N\times m}(0,1) \\ Y &\sim \left[I - \hat{W}\right]^{-1} X + \sigma_y N_{N\times m}(0,1) \end{aligned} \tag{5.37}$$

where $\sigma_x = 0.1$ controls the strength of the stimuli, $\sigma_y = 0.01$, and $N_{N\times m}(0,1)$ represents an $N \times m$ matrix of independent, standard Gaussian random variables. The ground truth weight matrix is generated by

$$\hat{W} \sim \frac{\sigma_w}{\sqrt{N}} N_{N\times N}(0,1). \tag{5.38}$$

Following Girko's circular law, the eigenvalues of $\hat{W}$ lie approximately within a circle of radius σ_w with high probability [33]. Hence, we take $\sigma_w = 0.5 < 1$ to control the spectral radius of the circle to be less than 1, so that all eigenvalues of the Jacobian matrix, $J = -I + W$, are negative and fixed point firing rates are stable.

We first study the explicit minimizers of this "over-parameterized" $(N > m)$ linear example. Note that, because the system is over-parameterized, the minimum cost is $J = 0$, which can be realized by infinitely many values of W. However, we must find a minimizer that produces stable fixed points. Stable fixed points are encouraged by W with a small spectral radius (because the Jacobian matrix is $J = -I + W$). We therefore find W^* by minimizing $\|W\|$ subject to $[I - W]^{-1}X = Y$, since small norm in W is to promote stable solution. This is equivalent to finding

$$\min\|W\| \text{ subject to } [I - W]\, Y = X$$

After rearranging terms, this can be written as

$$\min\|W\| \text{ subject to } WY = Y - X$$

Now the expression became a familiar "least squares" problem with respect to W, and we can solve directly for

$$W^* = [Y - X]\, Y^+ \tag{5.39}$$

where Y^+ is the Moore-Penrose pseudoinverse of Y (See Appendix A.6 for more details).

Similar to the 1D case in Section 5.4.1.1, the cost function in terms of W is

$$J(W) = \frac{1}{m}\sum_{i=1}^{m}\Big([I - W]^{-1}\mathbf{x}^i - \mathbf{y}^i\Big)^T\Big([I - W]^{-1}\mathbf{x}^i - \mathbf{y}^i\Big). \tag{5.40}$$

This cost landscape, $J(W)$, cannot easily be visualized as a function of W for $N > 1$ because W has N^2 dimensions, so even $N = 2$ would be difficult to visualize. To help visualize $J(W)$, we first plotted it on a random line segment passing through W^* in $\mathcal{R}^{N\times N}$. Specifically, we defined the parameterized function

$$W(t) = W^* + \frac{tp\sigma_w}{\sqrt{N}} Z \tag{5.41}$$

where $Z \in \mathcal{R}^{N\times N}$ has independent random entries drawn from a standard normal distribution, $\sigma_w = 0.5$ is the same parameter used to generate $\hat{W}$ described above, $p = 5$ scales the magnitude of the perturbation, and t was varied from -1 to 1 to create the visualization of $J(W(t))$ (Figure 5.5A). This corresponds to plotting $J(W)$ along a one-dimensional slice of the space $\mathcal{R}^{N\times N}$ on which W lives. Note that the true minimizer, $W = W^*$, is sampled when $t = 0$. Note also that, by Girko's circular law, W is very likely to produce unstable fixed points whenever $|t| > \frac{\sqrt{1-\sigma_w^2}}{\sigma_w p} \approx 0.346$ (see Appendix A.5). Figure 5.5A shows the resulting cost curve for five random values of Z with the blue dashed lines marking the stability boundary $|t| = \frac{\sqrt{1-\sigma_w^2}}{\sigma_w p}$. The cost is relatively well-behaved within the stability region but poorly conditioned outside of this region because of the singularities produced by the matrix inverses in Eq. (5.40). Specifically, when $|t| > 0.346$, the spectral radius of W is larger than 1 so some eigenvalues are near 1 in magnitude. As a result, the $[I - W]^{-1}$ in Eq. (5.40) can lead to very large values of $J(W)$ when $|t| > 0.346$.

To further visualize the loss landscape, we repeated the procedure above in two dimensions by sampling values of W from a random *plane* passing through W^*. Specifically, we defined the parameterized function

$$W(t_1, t_2) = W^* + \frac{p\sigma_w}{\sqrt{N}}(t_1 Z_1 + t_2 Z_2) \tag{5.42}$$

where $Z_1, Z_2 \in \mathcal{R}^{N\times N}$ have independent random entries drawn from a standard

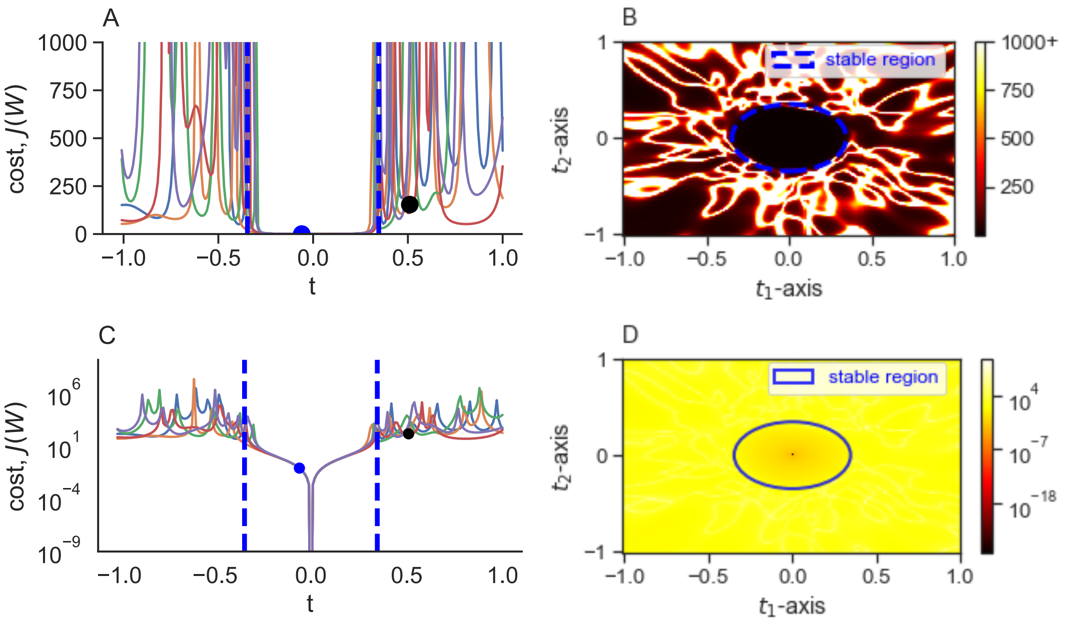

Figure 5.5. **Visualizing the cost landscape in a higher dimensional linear regression task. A)** The cost function $J(W(t))$ as a function of t from Eq. (5.41). This represents the cost evaluated along five random line segments in $\mathcal{R}^{N\times N}$, each passing through W^* at $t = 0$. Two blue dash lines show the stability boundary, $|t| = 0.346$. The vertical axis is cut off at $J = 1000$ to better visualize the curves. Blue and black circles show stable and unstable initial conditions used later for learning. **B)** The cost function $J(W(t_1, t_2))$ from Eq. (5.42). This represents the cost evaluated on a randomly oriented square with a center at W^*. The color axis is cut off at $J = 1000$.

normal distribution, $\sigma_w = 0.5$ is the same parameter used to generate $\hat{W}$ described above, $p = 5$ scales the magnitude of the perturbation, and t_1 and t_2 were each varied from -1 to 1 to create the visualization of $J(W(t))$ (Figure 5.5B). Note that $W(t_1, t_2) = W^*$ when $t_1 = t_2 = 0$, so the center of the square corresponds to the minimum cost, $J = 0$. Note also that the stability condition becomes $t_1^2 + t_2^2 < \frac{1-\sigma_w^2}{\sigma_w^2 p^2} = 0.12$ (see Appendix A.5), so the stability boundary is a circle (Figure 5.5B, dashed blue curve). Singularities create intricate ridges of large cost outside of the stability boundary (Figure 5.5B).

In summary, Figure 5.5 shows that the cost landscape is extremely poorly conditioned outside of the stability region, *i.e.*, when W have a spectral radius larger than 1.

5.4.2.1 Learning Through the Gradient Descent Approach

We first perform gradient descent on J with respect to W. The gradient-based update rule from Eq. (5.7) can be written as

$$\begin{aligned}\Delta W_1 &= \frac{1}{m}\sum_{i=1}^{m} \Delta W_1^i \\ &= \frac{-2\eta_W}{m}\left[I - W^T\right]^{-1}\left[R - Y\right] R^T.\end{aligned} \tag{5.43}$$

Following the discussion above, we hypothesized that gradient descent using initial values of W from the stable region (spectral radius < 1) would perform better compared to the initial values from the unstable region (spectral radius > 1).

Indeed, when the initial W was chosen from within the stability region, gradient descent approached the minimum cost, $J = 0$, after sufficiently long learning for sufficiently small learning rates (Figure 5.6A). In contrast, when the initial W was chosen from outside the stability region, W appears to get stuck in local minima and the cost remains high across a wide range of learning rates (Figure 5.6B).

Even for initial conditions within the stability region (Figure 5.6A), the effectiveness of gradient-based learning was somewhat sensitive to the choice of the learning rate, consistent with the observations we made for one-dimensional linear regression. We next show that the re-parameterized learning rule, ΔW_2^i, from Eqs. (5.10) and (5.17) improves learning robustness.

Under the stable fixed point condition compare to 1D case in Figure 5.2, $t < 1$ implies a negative Jacobian, which corresponds to a positive second derivative of Eq. (5.27), $J''(t) > 0$ for all w. Since we use σ_y to control the spread of the noise, convexity is guaranteed whenever the initial firing rates in Eq. (5.28) are not too far away from their target rates in Eq. (5.29) such that $\mathbf{y}_i < \frac{3}{2}\mathbf{r}_i$ in Figure 5.2B, so keeping the linear relationship satisfied. (see Appendix A.4). Hence performing the gradient descent method with a suitable learning rate, we can reach to the optimal cost.

However, in the higher dimensional case, it's hard to determine if the loss function associated with an initial random connectivity matrix W is still convex (Figure 5.5). This brings us to consider a convex cost function from reparameterization below.

5.4.2.2 Learning Under the Reparameterized Update.

For this linear case, the reparameterized learning rule from Eqs. (5.10) and (5.17) can be written as

$$\begin{aligned} \Delta W_2 &= \frac{1}{m}\sum_{i=1}^{m} \Delta W_2^i \\ &= [I - W] - \left([I - W]^{-1} - \frac{2\eta_A}{m}(R - Y)X^T\right)^{-1}. \end{aligned} \tag{5.44}$$

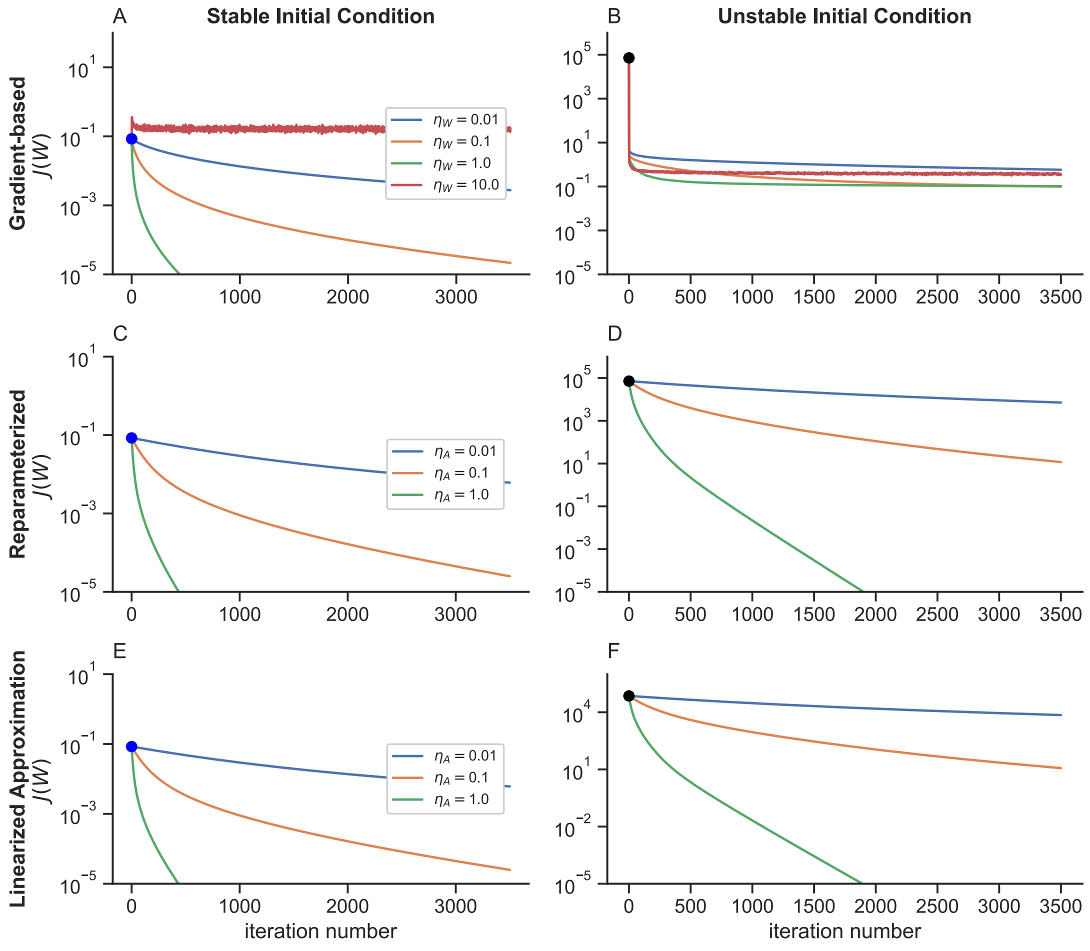

Figure 5.6. **Performance of three different learning rules for a linear regression problem in many dimensions.** Three different learning rules: ΔW_1^i (top row; A, B), ΔW_2^i (middle row; C, D), and ΔW_3^i (bottom row; E, F) applied to a linear regression problem in $N = 200$ dimensions with $m = 100$ data points, so the system is over-parameterized ($N > m$). In the left column (A, C, E) the initial W was drawn from within the stability region (spectral radius < 1; blue dot in Figure 5.5A). In the right column (B, D, F) the initial W was drawn from outside of the stability region (spectral radius > 1; black dot in Figure 5.5A).

Recall that the learning dynamics produced by Eq. (5.44) are equivalent to those produced by learning the standard cost function for a linear model,

$$J(A) = \frac{1}{m}\sum_{i=1}^{m}\left(A\mathbf{x}^i - \mathbf{y}^i\right)^T\left(A\mathbf{x}^i - \mathbf{y}^i\right). \tag{5.45}$$

along with the standard gradient-based update rule,

$$\Delta A = -\frac{2\eta_A}{m}(AX - Y)X^T. \tag{5.46}$$

In comparison to the gradient-based update, ΔW_1, on $J(W)$ from Eq. (5.43) (Figure 5.6A,B), we see that ΔW_2 from Eq. (5.44) performs much more robustly (Figure 5.6C,D). The cost converges toward the minimum at $J = 0$ across a range of learning rates whether the initial W is inside or outside the stability region. This is because the standard optimization problem defined by Eqs. (5.45) and (5.46) has robust convergence properties.

5.4.2.3 Learning Via Linearizing Reparameterized rule

We now consider the linearized reparameterized learning rule. The radius of convergence from Eq. (5.19) for this model can be written as

$$\eta_A < \frac{1}{\left\|\frac{2}{m}\left(XX^T - [I-W]\,YX^T\right)\right\|}, \tag{5.47}$$

The linearized, reparameterized update rule from Eq. (5.18) for this linear model can be written as

$$\begin{aligned}\Delta W_3 &= \frac{1}{m}\sum_{i=1}^{m}\Delta W_3^i \\ &= -\frac{2\eta_A}{m}\left[I-W\right](R-Y)X^T\left[I-W\right].\end{aligned} \tag{5.48}$$

With the trade-off of imposing the additional convergence criterion in Eq. (5.47), the approximated learning rule gives a cleaner form. The performance of this approximation update in Figure 5.6E and F shows excellent agreement with the reparameterized rule, ΔW_2 (Figure 5.6C-F). In terms of the computations, Eq. (5.48) does not require any explicit computation of matrix inverses with the exception of the computation of firing rates $R = [I - W]^{-1}X$. However, since R is left-multiplied by $[I - W]$, we can get rid of this matrix inverse to write ΔW_3 in the form

$$\Delta W_3 = -\frac{2\eta_A}{m}\left(XX^T\left[I - W\right] - \left[I - W\right]YX^T\left[I - W\right]\right). \tag{5.49}$$

Empirical tests show that using Eq. (5.49) for ΔW_3 is much faster than Eq. (5.43) for ΔW_1 or (5.44) for ΔW_2

Table 5.1 shows the runtime computations from stable (unstable) initial conditions within over and under-parameterized systems in seconds. Runtimes are for 3500 iterations of each learning rule across three different η values ($\eta = 0.01, 0.1, 1$) in a model with m samples and $N = 200$ neurons. To check the similarity between the updates from each learning rule, we calculated the angle between them, defined by

$$\begin{aligned}\theta_{\alpha,\beta} &= cos^{-1}\left(\frac{dW_\alpha \cdot dW_\beta}{\sqrt{(dW_\alpha \cdot dW_\alpha)(dW_\beta \cdot dW_\beta)}}\right)\\ &= cos^{-1}\left(\frac{Tr(dW_\alpha dW_\beta^T)}{\sqrt{Tr(dW_\alpha dW_\alpha^T)Tr(dW_\beta dW_\beta^T)}}\right).\end{aligned} \tag{5.50}$$

Here, α and $\beta = 1, 2, 3$ refer to the learning rules in Eqs. (5.43), Eqs. (5.44), and Eqs. (5.46). We use the β rule to update W throughout the comparison. Figure 5.7A, C, and E (left column) show the angle between the update rules through the learning process. We also computed the Pearson correlation coefficient between the updates,

TABLE 5.1

RUNTIME LEARNING RULE COMPARISON IN HIGHER DIMENSIONAL LINEAR LEAST SQUARES PROBLEM

System	Gradient-based Eq. (5.43)	Reparameterized Eq. (5.44)	Approximation Eq. (5.49)
Over parameterized ($N > m = 100$)	10.48(20.29)	21.49(29.30)	6.46(6.09)
Under-parameterized ($N < m = 500$)	25.18(25.45)	25.46 (42.31)	10.66 (22.13)

defined by

$$\rho_{\alpha,\beta} = \frac{cov(dW_\alpha, dW_\beta)}{\sqrt{var(dW_\alpha)var(dW_\beta)}}. \tag{5.51}$$

where *cov* and *var* are the covariance and variance taken across entries of the matrices. Figure 5.7B, D, and F (right column) show the correlation between the update rules through the learning process.

The re-parameterized and linear approximation updates (2 and 3) are very similar since the angle between them in Figure 5.7E is close to 0 and the correlation coefficient in Figure 5.7F is near 1. In contrast, the gradient-based update is less similar to the other two updates (Figure 5.7A-D).

5.4.2.4 An Under-parameterized Linear System.

For the sake of completeness, we also considered an under-parameterized system by increasing the number of training data points, m, to 500 while keeping the number of neurons at $N = 200$, so that $m > N$. We can continue using Eq. (5.39) and

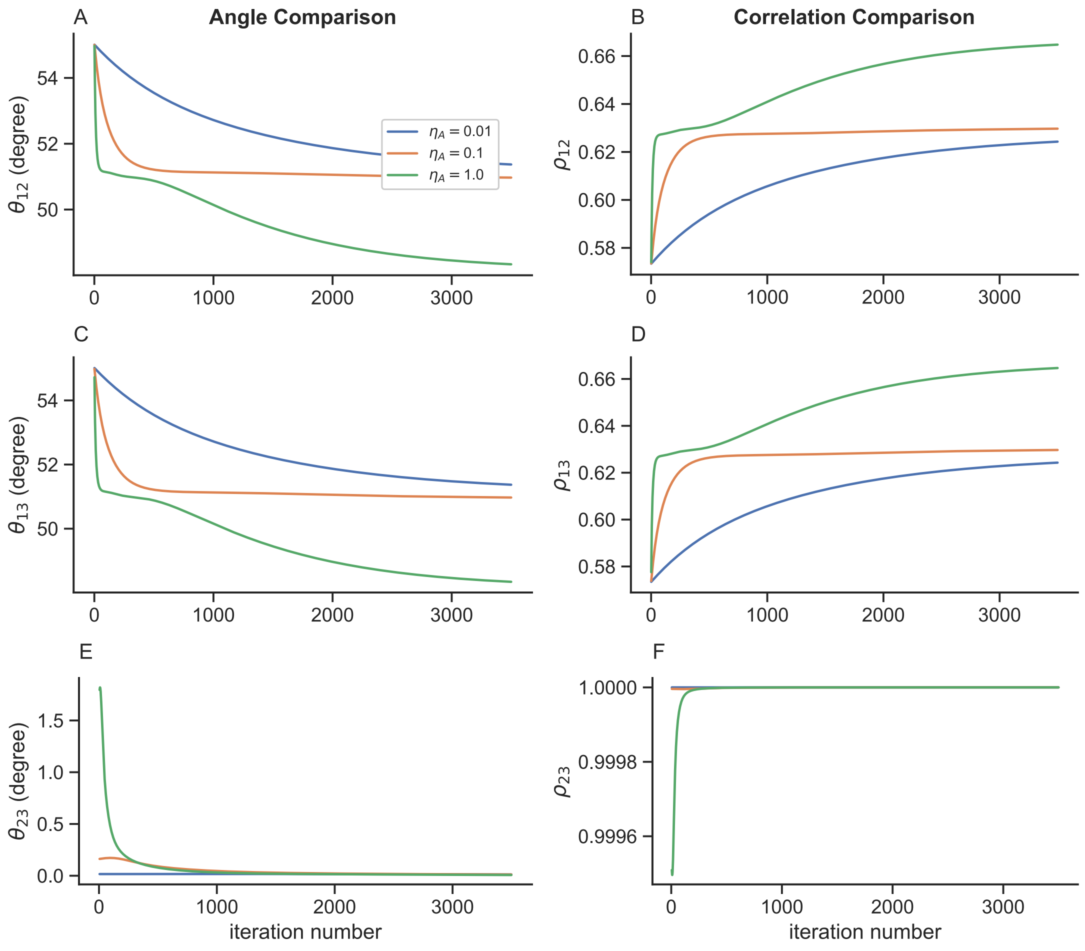

Figure 5.7. **Angles and correlations between weight updates for different learning rules. A)** Angle (θ_{12}) between the weight updates for the gradient-based and re-parameterized learning rules. **B)** Correlation coefficient (ρ_{12}) between the weight updates for the gradient-based (ΔW_1) and re-parameterized (ΔW_2) learning rules. **C-F)** Same as A and B, but for all other pairs of learning rules.

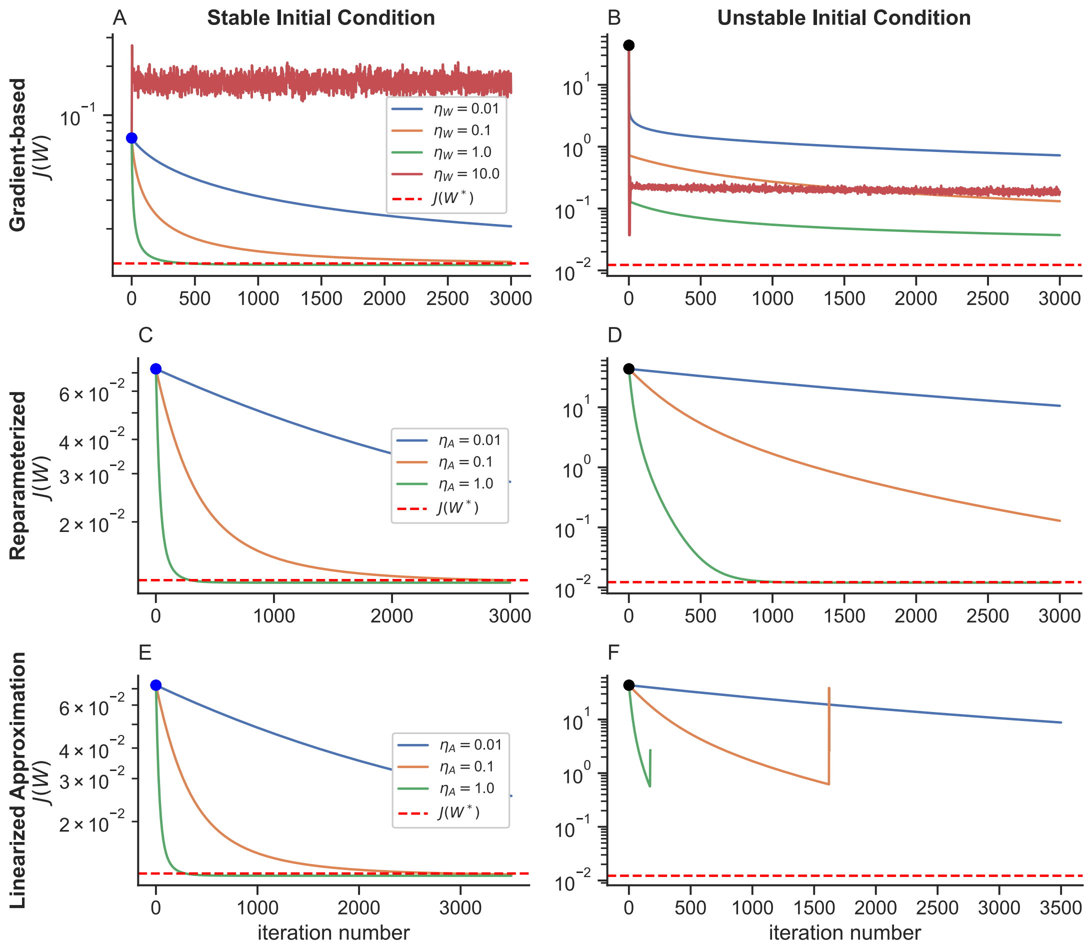

Figure 5.8. **Performance of three different learning rules for a linear regression problem in many dimensions using an under-parameterized model.** Same as Figure 5.6, but for an under-parameterized model with $m = 500 > N = 200$. The dashed red line indicates the true minimum cost, $J(W^*)$.

Eq. (5.34) as a solution of W^* and A^* (see Appendix A.6). Except for the sample size, m, we keep everything else the same as in the overparameterized model. Similarly, we also pick one initial W from the stable region and compare the three synaptic updating performances in Figure 5.8, our experiments show that this system takes fewer iterations to approach the optimal loss, but it is computationally slower in comparison to the over-parameterized model, presumably because m is larger (Table 5.1). For unstable initial conditions, the linearized rule blows up at larger learning rates (Figure 5.6F), but the results are otherwise similar to the over-paramterized case (compare Figures 5.6 and 5.8).

5.4.3 Example 3: Training Fixed Points on A Categorical Task.

We next consider a categorization task using the MNIST hand-written digit benchmark. We want to minimize a cross-entropy loss on $C = 10$ classes using the cross-entropy loss function with one-hot encoded labels (see Figure 5.9A). Specifically,

$$L(\mathbf{y}^i, \mathbf{s}^i) = -\mathbf{y}^i \cdot \log(\mathbf{s}^i) \tag{5.52}$$

where $\mathbf{y}^i$ is a "one-hot" encoded label for digit i. Specifically, $\mathbf{y}^i$ is a C-dimensional binary vector with 1 for the entry corresponding to the correct label and zeros in all other entries. Also,

$$\mathbf{s}^i_l = \frac{e^{\mathbf{z}^i_l}}{\sum\limits_{k=1}^{C} e^{\mathbf{z}^i_k}}, \tag{5.53}$$

is the softmax output, and $\mathbf{z}^i \in \mathcal{R}^C$ is a logit computed from a random projection of fixed point rates of a recurrent network. Specifically,

$$\mathbf{z}^i = W_{\text{out}}\mathbf{r}^i$$

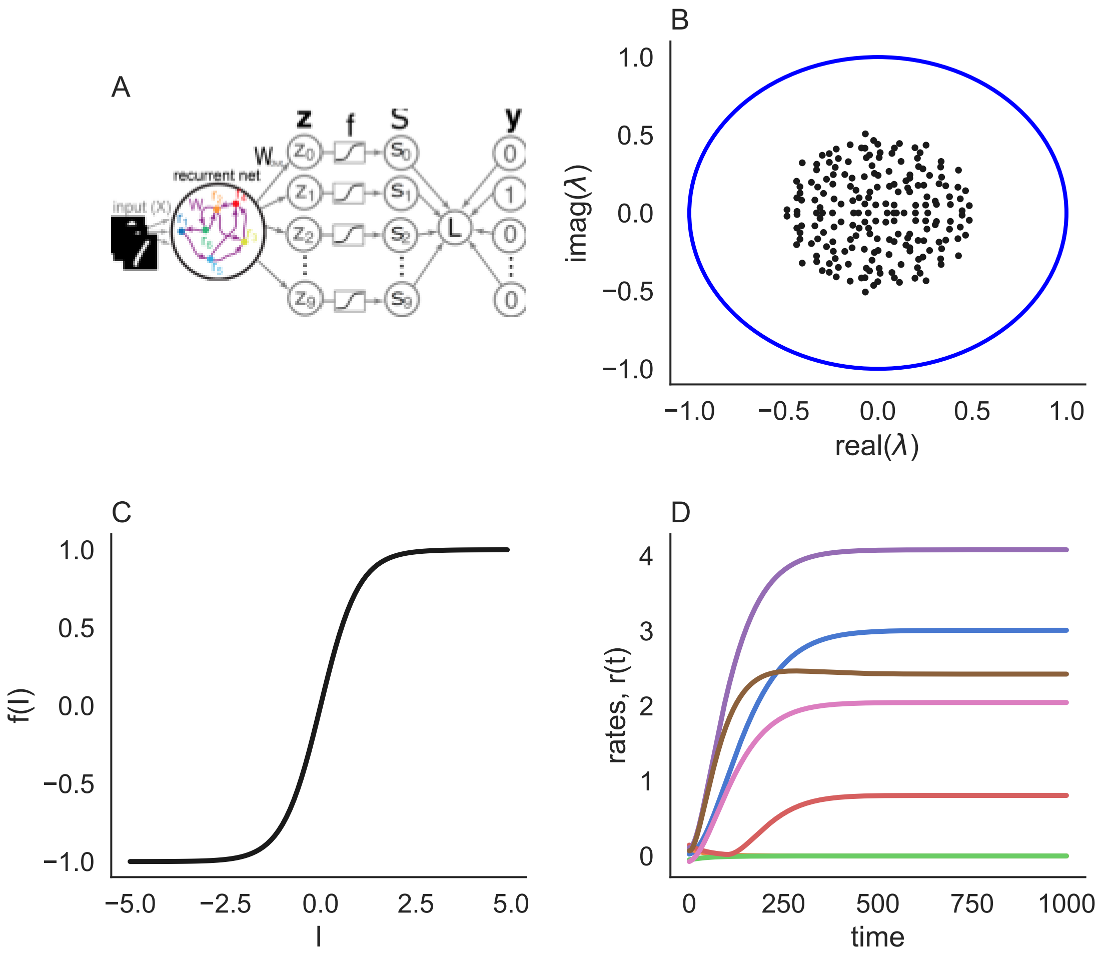

Figure 5.9. **Training fixed point from RNNs for a categorical task** **A**: Diagram of a RNN with MNIST-hand written digit inputs. **B)**: Eigenvalues of the connectivity matrix, W in the network, same as in Figure5.1B. **C**: Sigmoidal function used as an "f-I" curve, $f = tanh(I)$, where $G \neq Id$ in Eq.2.22. **D**: Same as in Figure5.1D, except the input is static image data.

where $W_{\text{out}} \in \mathcal{R}^{C\times N}$ is a fixed, random readout matrix and $\mathbf{r}^i = f(W\mathbf{r}^i + \mathbf{x}^i)$ is the fixed point from an $N \times N$ recurrent network with input i. Inputs are flattened 28×28 MNIST images, $\mathbf{p}^i \in \mathcal{R}^M$, where $M = 28 * 28 = 784$ and we multiply them by a fixed, random read-in matrix to form the input to the network,

$$\mathbf{x}^i = W_{\text{in}}\mathbf{p}^i$$

where $W_{\text{in}} \in \mathcal{R}^{N\times M}$ and $N = 200$ is the number of neurons in our network. We did not learn W_{out} or W_{in} because we wanted to focus on the effectiveness of learning the recurrent weight matrix, W. In all examples, below we train the network on $m = 100$ MNIST data points. To clarify, W_{out} and W_{in} are not learned, but they are held fixed so that we can evaluate learning of W alone.

Figure 5.9 presents the network scheme and the fixed point dynamics under the stability. In the following sections, we focus on training the fixed point with different activation functions including non-linear scenarios and compare the learning performance across with three different learning rules.

5.4.3.1 Learning with Linear Activation Function

We first consider the special case where $f(z) = z$ is the identity so that $G = I$. The matrix of firing rates $R = [\mathbf{r}^1 \;\; \mathbf{r}^2 \;\; \dots \;\; \mathbf{r}^m] \in \mathcal{R}^{N\times m}$ is given by $R = [I - W]^{-1}X$ where $X = [\mathbf{x}^1 \;\; \mathbf{x}^2 \;\; \dots \;\; \mathbf{x}^m] \in \mathcal{R}^{N\times m}$.

Gradient-based learning of W from Eq. (5.6) becomes

$$\begin{aligned} \Delta W_1 &= \frac{1}{m}\sum_{i=1}^{m} \Delta W_1^i \\ &= -\frac{\eta_W}{m}\left[I - W^T\right]^{-1} W_{\text{out}}^T \left[S - Y\right] R^T. \end{aligned} \tag{5.54}$$

where $Y = [\mathbf{y}^1 \;\; \mathbf{y}^2 \;\; \dots \;\; \mathbf{y}^m] \in \mathcal{R}^{C\times m}$ and $S = [\mathbf{s}^1 \;\; \mathbf{s}^2 \;\; \dots \;\; \mathbf{s}^m] \in \mathcal{R}^{C\times m}$ Similarly, the re-parameterized learning from Eq. (5.17) has the following expression,

$$\begin{aligned}\Delta W_2 &= \frac{1}{m}\sum_{i=1}^{m}\Delta W_2^i \\ &= -\left([I-W]^{-1} - \frac{\eta_A}{m} W_{\text{out}}^T [S-Y]\, X^T\right)^{-1} + [I-W]\,.\end{aligned} \tag{5.55}$$

Moreover, under the first order, Taylor expansion around small learning rate, η_A in Eq. (5.18) gives the synaptic update rule as,

$$\begin{aligned}\Delta W_3 &= \frac{1}{m}\sum_{i=1}^{m}\Delta W_3^i \\ &= -\frac{\eta_A}{m}[I-W]\, W_{\text{out}}^T [S-Y]\, X^T [I-W]\,.\end{aligned} \tag{5.56}$$

The gradient-based learning rule, ΔW_1^i, performs poorly across a wide range of learning rates. Small learning rates learn slowly while larger learning rates are unstable (Figure 5.10A, B). In contrast, the re-parameterized learning rule, ΔW_2, performs well across a range of learning rates (Figure 5.10C, D) and its linearized counterpart, ΔW_3^i, performs similarly (Figure 5.10E, F). Runtimes and testing errors for ΔW_3^i are greatly improved compared to ΔW_2^i and ΔW_1^i respectively (Table 5.2).

For the sake of completeness, we also considered an under-parameterized system by increasing m to 500 while keeping N = 200, so that $m > N$ (see Figure 5.11 and Table 5.2). We saw similar performance to the overparameterized case except in general the over-parameterized system has a higher testing error, presumably due to over-fitting.

Table 5.2 shows the computations of run time and errors for three different learning rules in an over (under) parameterized linear network with $m = 100(500)$ in training and testing samples on a categorical MNIST task across 500 iterations.

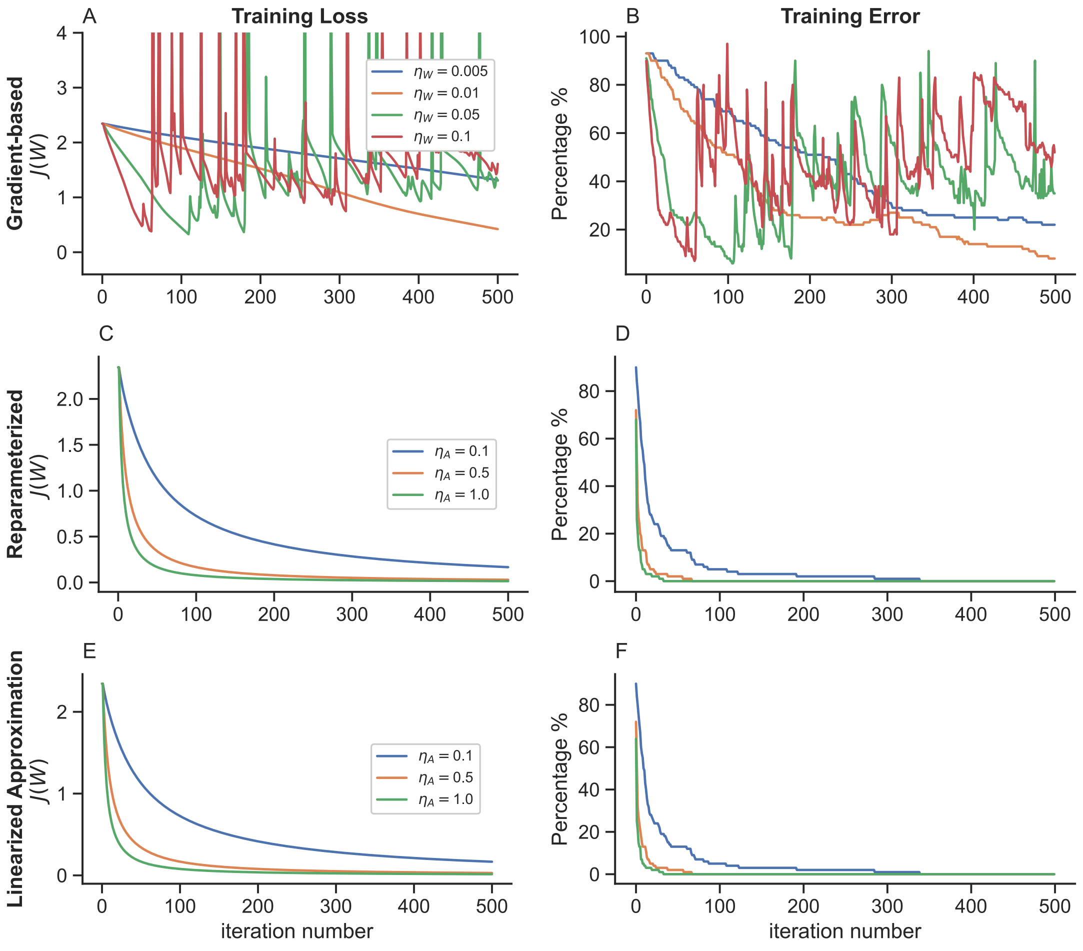

Figure 5.10. **Supervised linear learning of the fixed point firing rates on an over-parameterized categorical task. A)** Three different learning rules: ΔW_1^i in Eq. (5.55) (top row; A, B), ΔW_2^i in Eq. (5.54) (middle row; C, D), and ΔW_3^i in Eq. (5.56) (bottom row; E, F) applied to a categorical MNIST benchmark problem in a linear network of $N = 200$ dimensions with $m = 100$ data points, so the system is over-parameterized ($N > m$). The cost $J(W)$ is computed from the entropy loss in Eq. (5.52) across 500 iterations. The left column (A, C, E) is the training loss. Right Column (B, D, F)is the training error.

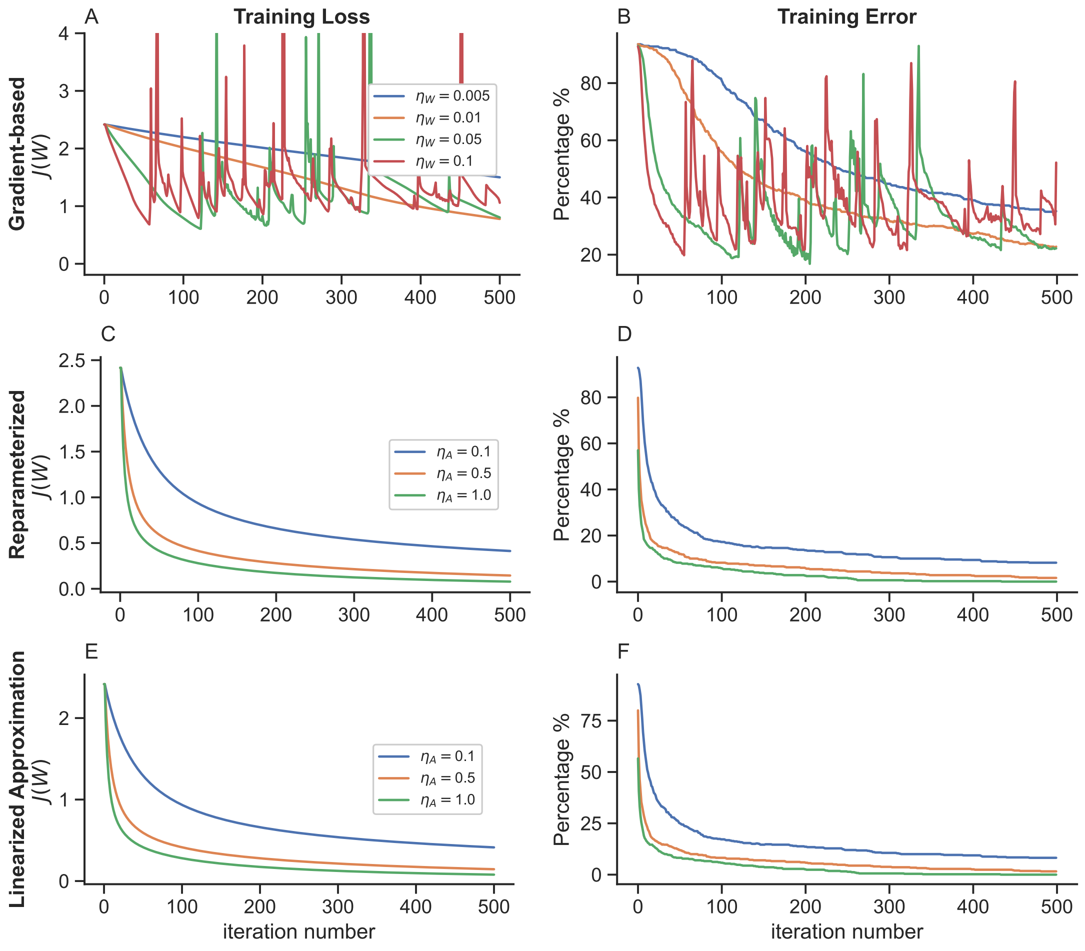

Figure 5.11. **Supervised linear learning of the fixed point firing rates on the under-parameterized categorical task.** Same as in Figure 5.10, except this is an under-parameterized system.

TABLE 5.2

LEARNING RULE RUNTIME AND ERROR COMPARISON OF A CATEGORICAL TASK VIA LINEAR LEARNING

Linear Learning Rule	Gradient-based Eq. (5.54)	Reparameterized Eq. (5.55)	Approximation Eq. (5.56)
Time (in *sec.)*	4.32(4.97)	22.79(24.59)	0.61(1.24)
Training Error (in %)	8.0(22.6)	0.0	0.0
Testing Error (in %)	40.0(36.6)	31.0(23.4)	31(23.4)

5.4.3.2 Learning with ReLu Activation using Support Sets

So far, we have only considered linear models. For our first test of a nonlinear model, we use a rectified linear activation, so that our networks become recurrent threshold-linear (a.k.s ReLu *or* Rectified Linear)recurrent networks [10, 23, 24]. Specifically, we used

$$f(z) = [z]^+ = \begin{cases} z & z > 0 \\ 0 & z \leq 0 \end{cases}. \tag{5.57}$$

Solving the fixed point equation in Eq. (5.3), then firing rates satisfy $\mathbf{r}^i = [W\mathbf{r}^i + \mathbf{x}^i]^+$. For $i = 1, \ldots, m$, we define a support set [23, 24] $S \subset \{1, \ldots, N\}$ corresponding to entries at which $\mathbf{r}^i$ is positive, whereas $S^c \subset \{1, \ldots, N\}$ is the complement set indicating indices at which $\mathbf{r}^i$ is zero. Then fixed points satisfy

$$\mathbf{r}^i_S = W_{S,S}\mathbf{r}^i_S + \mathbf{x}^i_S,$$

and

$$\mathbf{r}^i_{S^c} = 0.$$

Note that the support set is potentially different for each sample, i, so we should technically write S^i in place of S, but we just write S for notational convenience.

The gradient of $L(\mathbf{y}^i, \mathbf{s}^i)$ with respect to W is zero off-support. In other words,

$$\left[\nabla_W L(\mathbf{y}^i, \mathbf{s}^i)\right]_{jk} = \frac{\partial L}{\partial W_{jk}} = 0$$

if $j \in S^c$ or $k \in S^c$. Therefore, gradient descent with respect to W is the same as in a linear model from Eq. (5.6) but restricted to the support set (see Appendix A.7). Gradient-based updates can be written as

$$\Delta W_1^i = \begin{cases} -\eta_W \left[I_{S,S} - W_{S,S}^T\right]^{-1} \left[W_{\text{out}}\right]_{\cdot,S}^T \left(\mathbf{s}^i - \mathbf{y}^i\right) \left(\mathbf{r}_S^i\right)^T & \text{on } S \\ 0 & \text{on } S^c. \end{cases} \tag{5.58}$$

For updating synaptic connectivities through reparameterization, we must define the parameter over the support set, $A^i = \left[I_{S,S} - W_{S,S}\right]^{-1}$ and derive (see Appendix A.7),

$$\begin{aligned} \Delta A^i &= -\eta_A \left[W_{\text{out}}\right]_{\cdot,S}^T \left(\mathbf{s}^i - \mathbf{y}^i\right) \left(\mathbf{x}_S^i\right)^T \\ &= -\eta_A \left[W_{\text{out}}\right]_{\cdot,S}^T \left(\mathbf{s}^i - \mathbf{y}^i\right) \left(\mathbf{r}_S^i\right)^T \left[I_{S,S} - W_{S,S}^T\right] \end{aligned} \tag{5.59}$$

Then Eq. (5.17) becomes

$$\Delta W_2^i = \begin{cases} I_{S,S} - W_{S,S} - \left(\left[I_{S,S} - W_{S,S}\right]^{-1} - \eta_A \left[W_{\text{out}}\right]_{\cdot,S}^T \left(\mathbf{s}^i - \mathbf{y}^i\right) \left(\mathbf{x}_S^i\right)^T\right)^{-1} & \text{on } S \\ 0 & \text{on } S^c. \end{cases} \tag{5.60}$$

Linearizing this equation from Eq. (5.18) under the radius of convergence region in

TABLE 5.3

LEARNING RULE RUNTIME AND ERROR COMPARISON OF A CATEGORICAL TASK THROUGH RELU ACTIVATION WITH SUPPORT SETS

Rectified Learning Rule	Gradient-based Eq. (5.58)	Reparameterized Eq. (5.60)	Approximation Eq. (5.61)
Time (in *sec.)*	1266.55(5707.81)	2030.23(6441.73)	1074.21(5263.38)
Training Error (in %)	24.0(30.4)	0(11.4)	0(8.2)
Testing Error (in %)	52.0(40.6)	23.0(25.0)	33(27.6)

Eq. (5.19),

$$\Delta W_3^i = \begin{cases} -\eta_A \left[I_{S,S} - W_{S,S}\right] \left[W_{\text{out}}\right]_{\cdot,S}^T \left(\mathbf{s}^i - \mathbf{y}^i\right) \left(\mathbf{x}_S^i\right)^T \left[I_{S,S} - W_{S,S}\right] & \text{on } S \\ 0 & \text{on } S^c. \end{cases} \tag{5.61}$$

Fig. 5.12 compares the performance of these three learning rules. Note that gradient-based learning, ΔW_1^i, from Eq. (5.58) learns slowly for small learning rates, but blows up at larger learning rates, and is therefore not a robust learning algorithm for this task (Fig. 5.12A). The reparameterized learning rule and its linearized counterpart (ΔW_2^i and ΔW_3^i from Eqs. (5.60) and (5.61) respectively) perform much more robustly across a range of learning rates (Fig. 5.12C, D).

Note that the nonlinear computation time with the ReLu activation function in Table 5.3 takes much longer compared with the linear computation in Table 5.2 since we need to update each sample individually.

For completeness, we also computed the training loss and error for the rectified

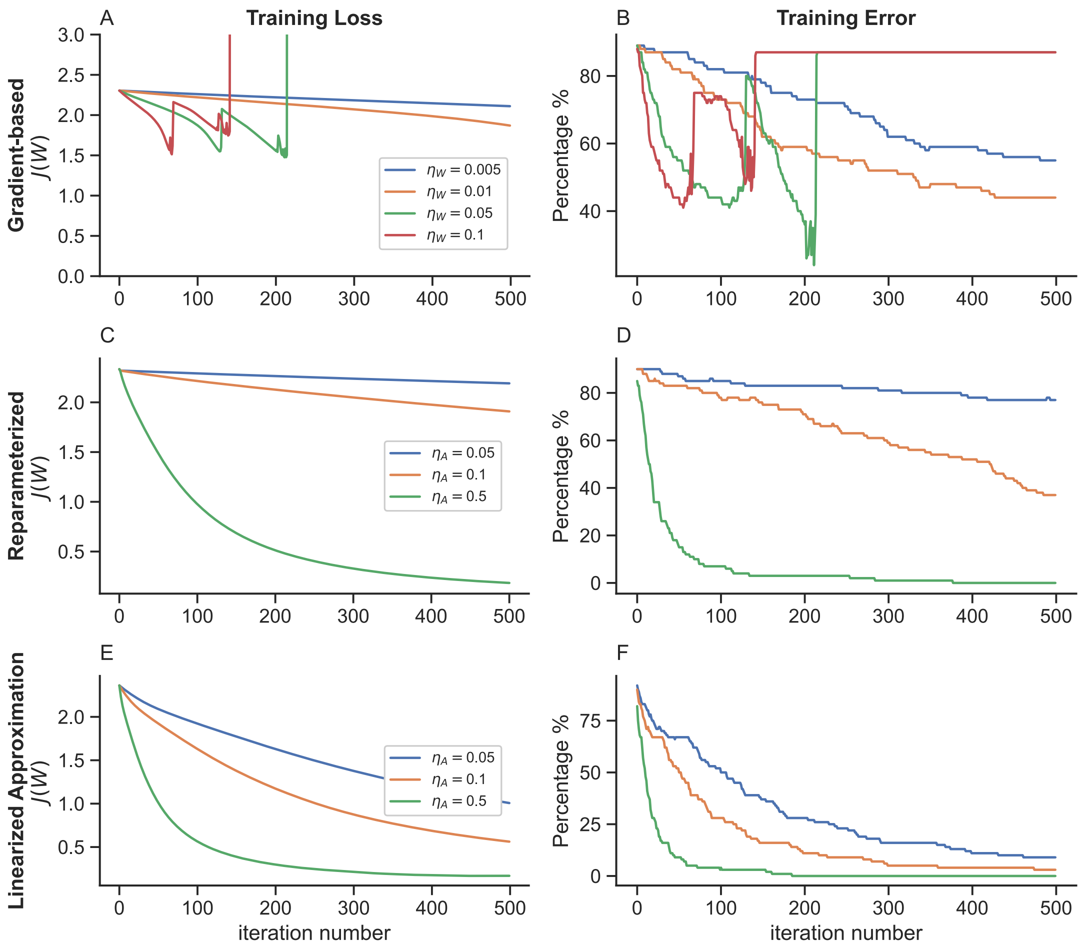

Figure 5.12. **Supervised learning of fixed point firing rates on a categorical task using a rectified linear activation function on a support set.** Same as in Fig. 5.10, except using the rectified linear activation from Eq. (5.57). Specifically, **A,B)** applied the learning rule from Eq. (5.58). **C,D)** used the learning rule from Eq. (5.60). **E,F)** performs the learning update from Eq. (5.61)

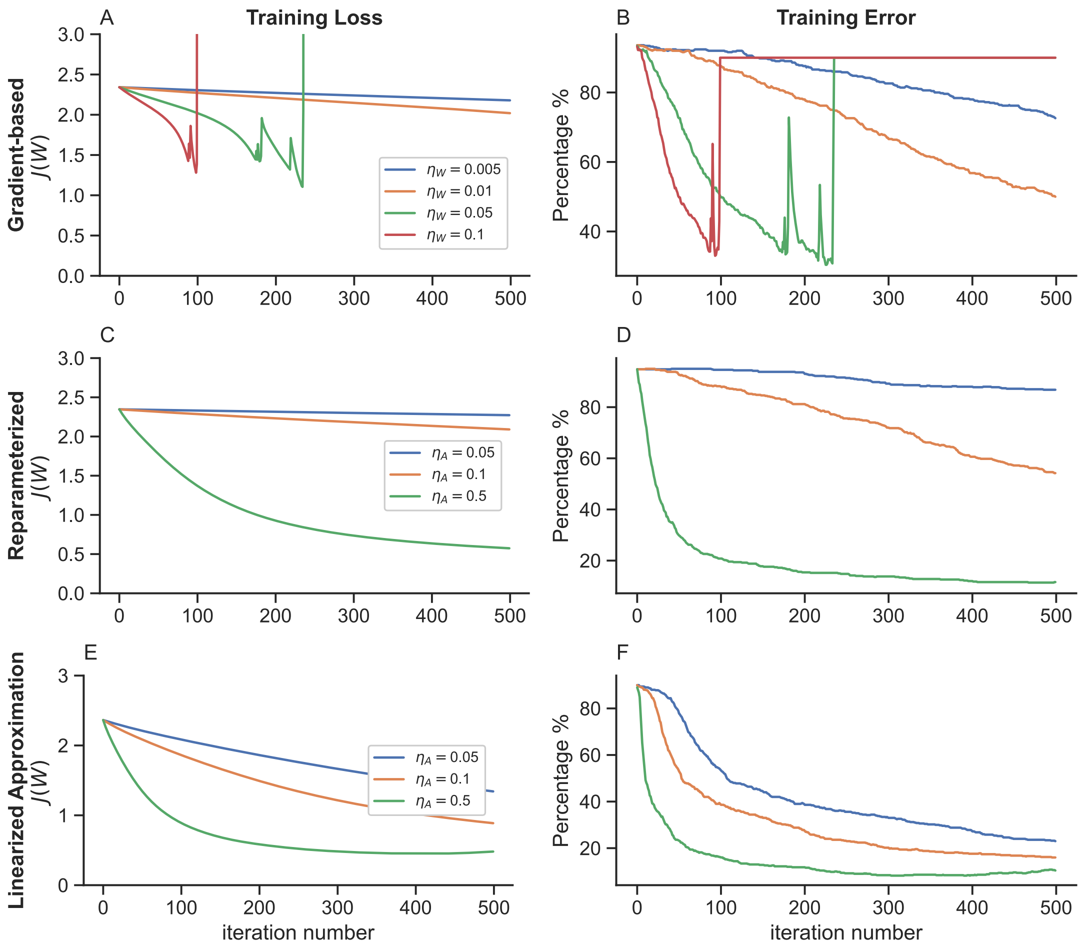

Figure 5.13. **Same as in Fig. 5.12 except this is an under-parameterized system.** With $m = 500$.

linear model within the under-parameterized case (Figure 5.13). The results were similar to the over-parameterized case.

5.4.3.3 Learning with a Sigmoidal Activation Function

Another type of nonlinear activation that is commonly used for RNNs in machine learning is the sigmoidal activation function. In particular, we use hyperbolic tangent

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{5.62}$$

Where the range has a value from -1 to 1.Instead of solving the fixed point equation in Eq. (5.3) analytically, for each sample, we use a numerical simulation approach to obtain the fixed point $\mathbf{r}^i$. Consequently, we now have different G^i. Then the gradient-based learning rule of W from Eq. (5.6) now becomes

$$\begin{aligned}\Delta W_1 &= \frac{1}{m}\sum_{i=1}^{m} \Delta W_1^i \\ &= -\frac{\eta_W}{m}\sum_{i=1}^{m} \left[\left[G^i\right]^{-1} - W^T\right]^{-1} W_{\text{out}}^T \left[\mathbf{s}^i - \mathbf{y}^i\right] \left(\mathbf{r}^i\right)^T .\end{aligned} \tag{5.63}$$

where G^i is also a diagonal matrix with diagonal entry $g_{jj}$is calculated from the the gradient of hyperbolic tangent $g_{jj} = 1 - \tanh^2(z)$. Since $\tanh(z)$ has previously stored, it is a faster and more efficient activation function in comparison to other sigmoid activation like the logistic function $f(z) = \frac{1}{1+e^{-z}}$. Similarly, the re-parameterized learning from Eq. (5.17) has the following expression,

$$\begin{aligned}\Delta W_2 &= \frac{1}{m}\sum_{i=1}^{m} \Delta W_2^i \\ &= -\sum_{i=1}^{m} \left(\left[\left[I - G^i W\right]^{-1} G^i - \frac{\eta_A}{m}[G^i]^2 W_{\text{out}}^T \left[\mathbf{s}^i - \mathbf{y}^i\right] (\mathbf{r}^i)^T \left[I - G^i W^T\right] G^i\right)^{-1} \\ &+ [G^i]^{-1} - W.\end{aligned} \tag{5.64}$$

Likewise, under the first order, Taylor expansion around small learning rate, η_A in Eq. (5.18) gives the synaptic update rule as,

$$\begin{aligned}\Delta W_3 &= \frac{1}{m}\sum_{i=1}^{m} \Delta W_3^i \\ &= -\frac{\eta_A}{m}\sum_{i=1}^{m} \left[I - W G^i\right] G^i W_{\text{out}}^T \left[\mathbf{s}^i - \mathbf{y}^i\right] (\mathbf{r}^i)^T \left[I - G^i W^T\right] \left[I - G^i W\right] .\end{aligned} \tag{5.65}$$

Figures 5.14 and 5.15 and Table 5.4 demonstrate similar results for the sigmoidal activation function as the rectified linear case. Notably, the gradient-based learn-

TABLE 5.4

LEARNING RULE RUNTIME AND ERROR COMPARISON OF A CATEGORICAL TASK THROUGH TANH ACTIVATION

tanh Learning Rule	Gradient-based Eq. (5.64)	Reparameterized Eq. (5.64)	Approximation Eq. (5.65)
Time (in *sec.)*	4535.66(8762.34)	3483.21(8968.40)	1318.59(5442.67)
Training Error (in %)	0.0	0.0	0.0
Testing Error (in %)	37.0(29.2)	35.0(31.0)	33.0(30.0)

ing, ΔW_1^i, performs worse than the reparameterized learning rule and its linearized counterpart (ΔW_2^i and ΔW_3^i).

5.5 Discussions

In summary, we have shown that when learning fixed points of recurrent neural network models, the direct application of gradient descent with respect to the recurrent weight matrix under Euclidean geometry is computationally expensive and not robust. Badly conditioned loss surfaces and singularities in the loss surfaces can cause ineffective learning. Matrix inverses in the equations for the gradients are expensive to evaluate.

We proposed two alternative learning rules derived from a reparameterization of the recurrent network model. These learning rules perform more robustly than the standard gradient descent approach. Moreover, one of the two learning rules is computationally much more efficient. The learning rules can be interpreted as steepest descent and gradient descent on the recurrent weight matrix under a non-Euclidean metric. Our results support recent calls to re-consider the default use of

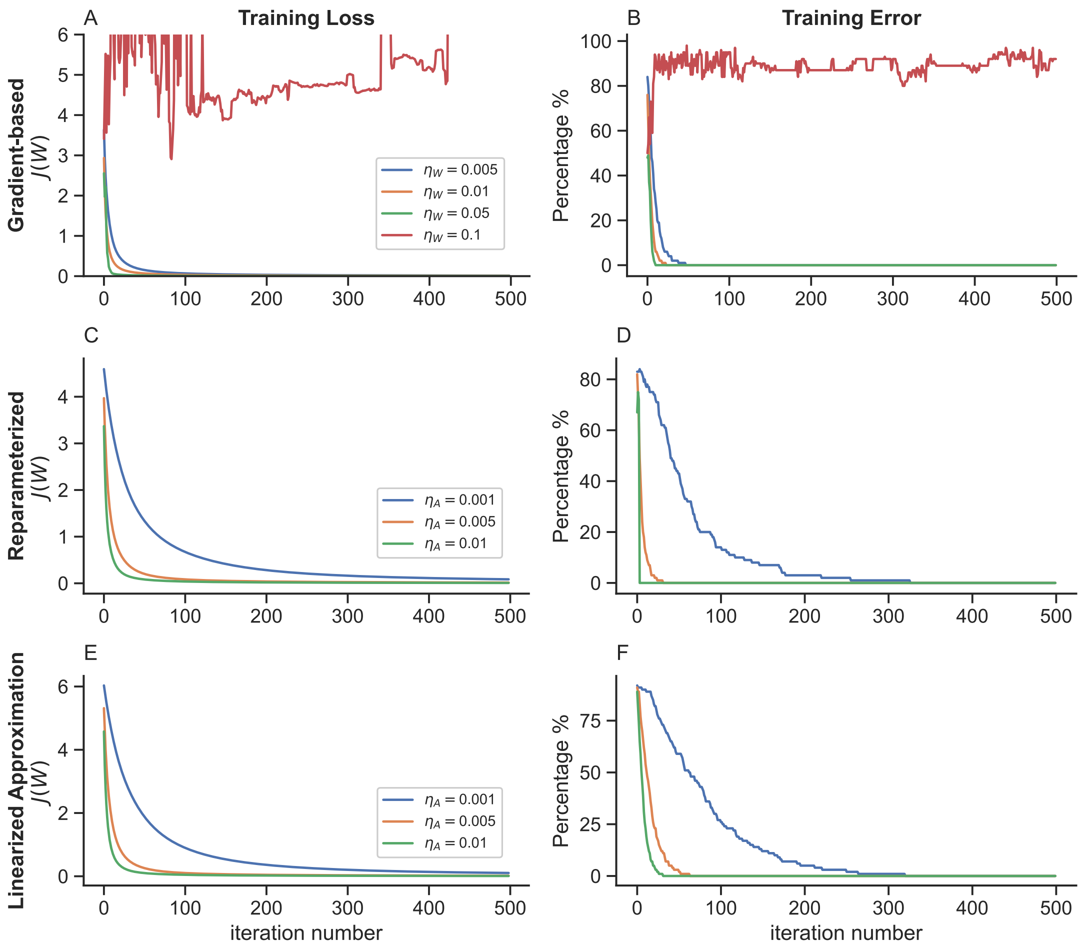

Figure 5.14. **Supervised learning of fixed point firing rates on a categorical task using a sigmoidal activation function.** Same as in Fig. 5.10 and Fig. 5.12, except using the tanh activation from Eq. (5.62). Specifically, **A,B)** applied the learning rule from Eq. (5.64). **C,D)** used the learning rule from Eq. (5.64). **E,F)** performs the learning update from Eq. (5.65).

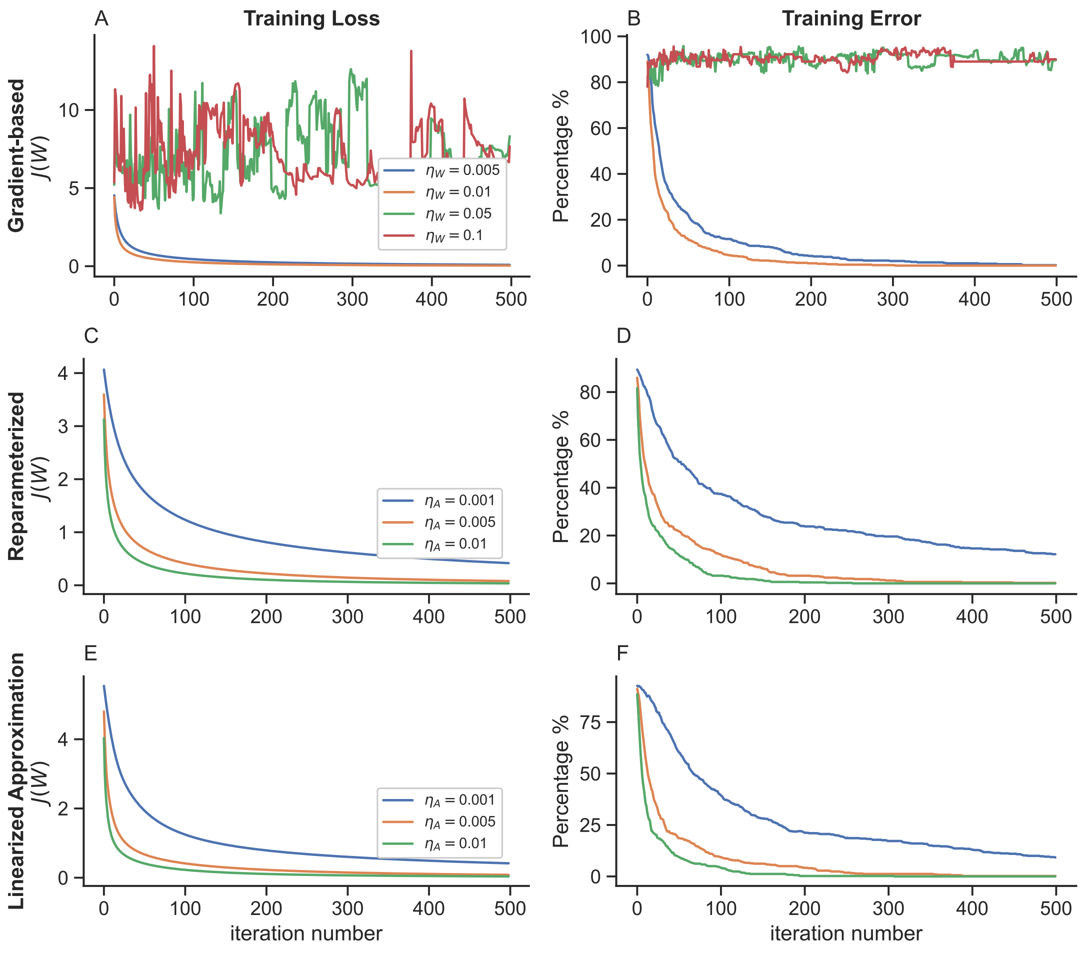

Figure 5.15. **Same as in Fig. 5.14 except this is an under-parameterized system.** With $m = 500$.

Euclidean gradients on parameters in machine learning [5, 6, 59] and computational neuroscience [78]. These results also have implications for training recurrent neural network models for computational neuroscience research and for machine learning applications.

CHAPTER 6

SUMMARY AND FUTURE DIRECTIONS

We start the brain-modeling Odyssey with a discovery of neural dynamics. We use the EIF spiking model to describe neural activities and their biological features in-depth. Inspired by the neuron firing responses, a simplified mean-field approximation of rate models and their connections to the artificial neural framework in machine learning were developed, which allows us to analyze other computational properties like the balance network, stimulus representations, and synaptic plasticity. Along with the model evolution, they become more capable to solve real tasks such as error detection and learning some objects under supervision. As a trade-off, simplified models also become more abstract and drift away from biological details and explanations. Studying the models of recurrent neuronal networks has brought us a wealth of knowledge to understand neural dynamics and computations.

6.1 Summary

We summarize our learning objectives through each chapter and show their interweaving connections here.

Recurrent Neural Networks. The objective of studying this whole area is to advance the knowledge in order to understand the biological layer of the cortical neural activities and computations in the human brain. We start with a brief review of the basic biological structures of the brain and neurons in chapter 1. We also highlight their differences and connections with artificial neural networks in machine learning. We then discuss how information is received and propagated in the brain as neurons communicate through synapses. Next in chapter 2, we use several spiking models to describe the biological action potential phenomenon from a single neuron to a network of neurons. This helps us to understand neuron firing rate dynamics and other biological properties such as Dale's principle, synaptic plasticity, "f-I" curve, excitatory-inhibitory balance, and many other computational properties in the recurrent neuronal networks.

Semi-balanced Network. One of the properties of biological neural networks is the input cancellation balance of strong excitatory and inhibitory currents, which results in a moderate total input. Our first contribution in Chapter 3 is to show that every balanced network architecture admits stimuli that break the balanced state and these breaks in balance push the network into a "semi-balanced state" characterized by excess inhibition to some neurons, but an absence of excess excitation. In addition, we prove the semi-balanced state is equivalence to bounding rates and this is biologically realistic since it describes the general properties of strongly coupled networks with moderate firing rates *in vivo*. Finally, we establish a direct relationship between semi-balanced networks and artificial recurrent neural networks with rectified linear activation used in machine learning.

Homeostatic Plasticity Learns to Compute Prediction Errors. Beyond the discovery of models to describe neural biological properties and computations in the recurrent networks, we then focus on understanding the model capabilities in the recurrent neuronal networks that can actually produce something. One of the model abilities involves learning. In chapter 4, we designed "prediction errors" as mismatched stimuli pairs and evaluate whether the inhibitory synaptic plasticity rules can learn to compute prediction errors in unstructured neural networks. We also verify the accuracy of the rate model through mean-field theory and show that it can be not only used to approximate spiking models but can also use to explain the learning performance. We find that homeostatic plasticity is sufficient to compute prediction errors for trivial time-constant stimuli, but not for more realistic time-varying stimuli. Despite some failures of our model detection, this work indeed helps us understand that with plasticity the network structure is crucial for learning predictive coding tasks.

Learning Fixed Point in Recurrent Neuronal Networks. Previous chapters have outlined the connections between the rectified activation in artificial neural networks and the fixed point of firing rate models in biological neural networks. We then omit the biological details of spiking and start with a recurrent rate network model in chapter 5. We train the fixed points in our recurrent neuronal networks and perform several supervised learning tasks through different learning algorithms. In addition to the traditional gradient descent approach on the recurrent weight matrix, we propose new learning rules under a natural re-parameterization of the networks and their approximation to the linear order. We show that the naïve gradient descent method is computationally expensive and leads to poor learning performance for both the higher dimensional linear least square problems and the categorical tasks using the MNIST benchmark. In contrast, our re-parameterized learning rules can overcome these obstacles and produce more efficient and robust results.

6.2 Future Work

My work so far has brought two perspectives on the realism of recurrent neural networks. One is to link neuroscience and machine learning by establishing a direct, one-to-one analogue between artificial and biological neuronal networks. The other one is to develop new learning rules for training the recurrent neuronal network responses and perform a wide range of supervised learning tasks.

One future direction is to continue building the connections between artificial and biological recurrent neural networks by extending the results from a single-layered recurrent neuronal network to multi-layered *and/or* deep recurrent neuronal networks, which is more biologically and realistically linked to cortical circuits. The neuronal network can learn static functions as well as functions between time series. Multi-layered recurrent networks in machine learning are expensive to train using back-propagation through time, but synaptic plasticity does not have this problem. I will continue the biological connections with plasticity learning and also account for other efficient machine learning algorithms. Specifically, short-term synaptic plasticity introduces nonlinearities, and together with the nonlinearities introduced by the activation functions in machine learning, these results could help account for the long-term dependencies in the recurrent neuronal networks.

Another direction is to continue developing more efficient learning algorithms in recurrent neuronal networks by including the capabilities of solving more difficult tasks while still biologically realistic. For example, I would like to extend our model capacity to unsupervised learning tasks such as clustering and association. In addition, I would like to discover other machine learning approaches like reinforcement learning through agency and rewards. Furthermore, I would like to connect our biological recurrent neural network models with operational research and develop decision-making tasks where I can demonstrate the model improvements and its explainable potential with artificial intelligence in the real business applications.

Beyond the current work direction on learning, the brain-modeling Odyssey will continue expanding in many possible directions as below:

- Use a bayesian framework to describe the sensory cortex and understand how the brain interacts with our "real" world through predictive coding.
- Take a causal inference approach to analyze the biological neural activities when a cause of a change in internal or external condition.
- Use more refined mathematical models such as introducing higher order *and/or* partial differential equations, which include space and time components to describe neural activities.
- Implement stochastic elements for the biological recurrent network. Stochastic input and output simulations are useful to understand neural computations.

Likewise, the stochastic components can also be useful in improving learning performance such as the stochastic gradient descent method can get rid of stock in the local landscape.

- Discover other biological properties other than sensory processing such as motor control and memory.
- Apply higher-order statistical description other than the first moment (*i.e.*, mean *or* expectation) such as using covariance or distribution to draw the connection between neuron features with other characteristics.
- Develop statistical hypothesis tests and derive estimates for inference and draw conclusions about neuron activities.
- ...

There are so many directions and methodologies that can be useful, and I will continue to describe the new problems as the solution has been created through a proposed project. Just like in architecture, the development of the arch allowed for the challenge of building cathedrals. So the developments in my previous work lay the groundwork for the problems of my future brain-modeling discovery.

APPENDIX A

LEARNING FIXED POINTS IN RNNS (CORRESPONDING TO CHAPTER 5)

A.1 Derivation of the Direct Gradient Descent Rule

Here, we derive Eq. (5.6) for direct gradient descent on W. To derive Eq. (5.6), it is sufficient to show that

$$\nabla_W L(\mathbf{r}^i(W)) = \left(\mathbf{r}^i \left[\nabla_{\mathbf{r}^i} L\right]^T \left[I - G^i W\right]^{-1} G^i\right)^T.$$

We know $\nabla_W L(\mathbf{r}^i(W))$ is a matrix with elements

$$\frac{\partial L}{\partial W_{jk}} = \left[\nabla_{\mathbf{r}^i} L(\mathbf{r}^i, \mathbf{y}^i)\right] \cdot \frac{\partial \mathbf{r}^i}{\partial W_{jk}},$$

To derive $\frac{\partial \mathbf{r}^i}{\partial W_{jk}}$, we first derive the change of firing rate, $\Delta \vec{r^i}$ to linear order in ΔW_{jk},

$$\begin{aligned}
\Delta \mathbf{r}^i &= f(\mathbf{z}) - f(\mathbf{z}_0) \qquad\qquad \text{by Taylor expansion} \\
&= f(\mathbf{z}_0) + \frac{f'(\mathbf{z}_0)}{1!}(\mathbf{z} - \mathbf{z}_0) - f(\mathbf{z}_0) + O(\mathbf{z} - \mathbf{z}_0)^2 \\
&= G^i(\mathbf{z} - \mathbf{z}_0) + O(\mathbf{z} - \mathbf{z}_0)^2.
\end{aligned}$$

Where $W = W_0 + \Delta W$, and $\mathbf{z}_0 = W_0\vec{r} + \vec{x}^i$ is the total input from the previous step. To linear order in $\Delta\mathbf{r}^i$, we have

$$\begin{aligned}
\Delta\mathbf{r}^i &= G^i(\mathbf{z} - \mathbf{z}_0) \\
&= G^i\left((W\mathbf{r}^i + \mathbf{x}^i) - (W_0\mathbf{r}_0^i + \mathbf{x}_0^i)\right) \quad = G^i\left((W_0 + \Delta W)\mathbf{r}^i - W_0\mathbf{r}_0^i\right) \\
&= G^i(W_0\mathbf{r}^i + \Delta W\mathbf{r}^i - W_0\mathbf{r}_0^i) \\
&= G^i(W_0\Delta\mathbf{r}^i + \Delta W\mathbf{r}^i) \\
\Delta\mathbf{r}^i - G^iW_0\Delta\mathbf{r}^i &= G^i\Delta W\mathbf{r}^i \\
[I - G^iW_0]\Delta\mathbf{r}^i &= G^i\Delta W\mathbf{r}^i.
\end{aligned}$$

As a result, we have that

$$\frac{\partial\mathbf{r}^i}{\partial W_{jk}} = [I - G^iW]^{-1}G^i\mathbf{1}_{jk}\mathbf{r}^i$$

which is interpreted as a column vector.

We first need to derive the derivative of $\vec{r}^i$ with respect to a single entry in W. To achieve this, We can first derive the change of firing rate in the linear order after an update to W,

$$\begin{aligned}
\Delta\mathbf{r}^i &= f(\mathbf{z}) - f(\mathbf{z}_0) \qquad\qquad \text{by Taylor expansion} \\
&= f(\mathbf{z}_0) + \frac{f'(\mathbf{z}_0)}{1!}(\mathbf{z} - \mathbf{z}_0) - f(\mathbf{z}_0) + O(\mathbf{z} - \mathbf{z}_0)^2 \\
&= G^i(\mathbf{z} - \mathbf{z}_0) + O(\mathbf{z} - \mathbf{z}_0)^2.
\end{aligned}$$

Where $\mathbf{z}_0$ is the total input from the previous step. To linear order in $\Delta\mathbf{r}^i$, we have

$$
\begin{aligned}
\Delta \mathbf{r}^i &= G^i(\mathbf{z} - \mathbf{z}_0) \\
&= G^i\left((W\mathbf{r}^i + \mathbf{x}^i) - (W_0\mathbf{r}_0^i + \mathbf{x}_0^i)\right) && \text{for constant input, } \mathbf{x}^i = \mathbf{x}_0^i \\
&= G^i\left((W_0 + \Delta W)\mathbf{r}^i - W_0\mathbf{r}_0^i\right) \\
&= G^i(W_0\mathbf{r}^i + \Delta W\mathbf{r}^i - W_0\mathbf{r}_0^i) \\
&= G^i(W_0\Delta\mathbf{r}^i + \Delta W\mathbf{r}^i) \\
\Delta\mathbf{r}^i - G^iW_0\Delta\mathbf{r}^i &= G^i\Delta W\mathbf{r}^i \\
[I - G^iW_0]\Delta\mathbf{r}^i &= G^i\Delta W\mathbf{r}^i.
\end{aligned}
$$

As a result, we have that

$$
\frac{\partial \mathbf{r}^i}{\partial W_{jk}} = [I - G^iW]^{-1}G^i\mathbf{1}_{jk}\mathbf{r}^i
$$

where $\frac{\partial \mathbf{r}^i}{\partial W_{jk}}$ is interpreted as a column vector. Eq. (5.6) then follows from the following Lemma.

Lemma 1 $[I - G^iW]^{-1}G^i\boldsymbol{1}_{jk}\boldsymbol{r}^i = \boldsymbol{r}_k^i\,[[I - G^iW]^{-1}G^i]_{(:,j)}$

Proof: We first calculate $\mathbf{1}_{11}\mathbf{r}^i$, $\mathbf{1}_{12}\mathbf{r}^i$, and $\mathbf{1}_{21}\mathbf{r}^i$:

$$
\mathbf{1}_{11}\mathbf{r}^i = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ 0 & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} \mathbf{r}_1^i \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{r}_M^i \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1^i \\ 0 \\ \cdot \\ \cdot \\ 0 \end{bmatrix} = \mathbf{r}_1^i I(:,1)
$$

$$\mathbf{1}_{12}\mathbf{r}^i = \begin{bmatrix} 0 & 1 & \dots & 0 \\ 0 & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ 0 & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} \mathbf{r}^i_1 \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{r}^i_M \end{bmatrix} = \begin{bmatrix} \mathbf{r}^i_2 \\ 0 \\ \cdot \\ \cdot \\ 0 \end{bmatrix} = \mathbf{r}^i_2 I(:,1)$$

$$\mathbf{1}_{21}\mathbf{r}^i = \begin{bmatrix} 0 & 0 & \dots & 0 \\ 1 & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ 0 & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} \mathbf{r}^i_1 \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{r}^i_M \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{r}^i_1 \\ \cdot \\ \cdot \\ 0 \end{bmatrix} = \mathbf{r}^i_1 I(:,2).$$

Denote $A := [I - G^iW]^{-1}G^i$, so $A\mathbf{1}_{11}\mathbf{r}^i = \mathbf{r}^i_1 A_{(:,1)}$, $A\mathbf{1}_{12}\mathbf{r}^i = \mathbf{r}^i_2 A_{(:,1)}$, and $A\mathbf{1}_{21}\mathbf{r}^i = \mathbf{r}^i_1 A_{(:,2)}$. Notice that they are column vectors. WLOG, $A\mathbf{1}_{jk}\mathbf{r}^i = \mathbf{r}^i_k A_{(:,j)}$

$$LHS = \nabla_w L(\mathbf{r}^i(W)) = \begin{bmatrix} \frac{dL}{dW_{11}} & \frac{dL}{dW_{12}} & \dots & \dots & \frac{dL}{dW_{1M}} \\ \frac{dL}{dW_{21}} & \frac{dL}{dW_{22}} & \dots & \dots & \frac{dL}{dW_{2M}} \\ \frac{dL}{dW_{j1}} & \dots & \frac{dL}{dW_{jk}} & \dots & \frac{dL}{dW_{jM}} \\ \frac{dL}{dW_{M1}} & \frac{dL}{dW_{M2}} & \dots & \dots & \frac{dL}{dW_{MM}} \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{r}^i_1[\nabla_{\mathbf{r}^i} L(\mathbf{r}^i)] \cdot A_{(:,1)} & \dots & \mathbf{r}^i_M[\nabla_{\mathbf{r}^i} L(\mathbf{r}^i)] \cdot A_{(:,1)} \\ \mathbf{r}^i_1[\nabla_{\mathbf{r}^i} L(\mathbf{r}^i)] \cdot A_{(:,2)} & \dots & \mathbf{r}^i_M[\nabla^i_{\mathbf{r}} L(\mathbf{r}^i)] \cdot A_{(:,2)} \\ \dots & \mathbf{r}_k[\nabla_{\mathbf{r}^i} L(\mathbf{r}^i)] \cdot A_{(:,j)} & \dots \\ \mathbf{r}^i_1[\nabla_{\mathbf{r}^i} L(\mathbf{r}^i)] \cdot A_{(:,M)} & \dots & \mathbf{r}^i_M[\nabla^i_{\mathbf{r}} L(\mathbf{r}^i)] \cdot A_{(:,M)} \end{bmatrix},$$

$$RHS = \left(\mathbf{r}^i[\nabla_{\mathbf{r}^i}L(\mathbf{r}^i)]^T[I - G^iW]^{-1}G^i\right)^T = \left(\mathbf{r}^i[\nabla_{\mathbf{r}^i}L(\mathbf{r}^i)]^T A\right)^T$$

$$= \left(\begin{bmatrix} \mathbf{r}_1^i\frac{\partial L(\mathbf{r}^i)}{\partial \mathbf{r}_1^i} & \mathbf{r}_1^i\frac{\partial L(\mathbf{r}^i)}{\partial \mathbf{r}_2^i} & \dots & \mathbf{r}_1^i\frac{\partial L(\mathbf{r}^i)}{\partial \mathbf{r}_M^i} \\ \mathbf{r}_2^i\frac{\partial L(\mathbf{r}^i)}{\partial \mathbf{r}_1^i} & \mathbf{r}_2^i\frac{\partial L(\mathbf{r}^i)}{\partial \mathbf{r}_2^i} & \dots & \mathbf{r}_2^i\frac{\partial L(\mathbf{r}^i)}{\partial \mathbf{r}_M^i} \\ \dots & \dots & \dots & \dots \\ \mathbf{r}_M^i\frac{\partial L(\mathbf{r}^i)}{\partial \mathbf{r}_1^i} & \mathbf{r}_M^i\frac{\partial L(\mathbf{r}^i)}{\partial \mathbf{r}_2^i} & \dots & \mathbf{r}_M^i\frac{\partial L(\mathbf{r}^i)}{\partial \mathbf{r}_M^i} \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1M} \\ A_{21} & A_{22} & \dots & A_{2M} \\ \dots & \dots & \dots & \dots \\ A_{M1} & A_{M2} & \dots & A_{MM} \end{bmatrix}\right)^T$$

$$= \begin{bmatrix} \mathbf{r}_1^i[\nabla_{\mathbf{r}^i}L(\mathbf{r}^i)]^T A_{(:,1)} & \mathbf{r}_1^i[\nabla_{\mathbf{r}^i}L(\mathbf{r}^i)]^T A_{(:,2)} & \dots & \mathbf{r}_1^i[\nabla_{\mathbf{r}^i}L(\mathbf{r}^i)]^T A_{(:,M)} \\ \mathbf{r}_2^i[\nabla_{\mathbf{r}^i}L(\mathbf{r}^i)]^T A_{(:,1)} & \mathbf{r}_2^i[\nabla_{\mathbf{r}^i}L(\mathbf{r}^i)]^T A_{(:,2)} & \dots & \mathbf{r}_2^i[\nabla_{\mathbf{r}^i}L(\mathbf{r}^i)]^T A_{(:,M)} \\ \dots & \dots & \dots & \dots \\ \mathbf{r}_M[\nabla_{\mathbf{r}^i}L(\mathbf{r}^i)]^T A_{(:,1)} & \mathbf{r}_M[\nabla_{\mathbf{r}^i}L(\mathbf{r}^i)]^T A_{(:,2)} & \dots & \mathbf{r}_M[\nabla_{\mathbf{r}^i}L(\mathbf{r}^i)]^T A_{(:,M)} \end{bmatrix}^T$$

$$= \begin{bmatrix} \mathbf{r}_1^i[\nabla_{\mathbf{r}^i}L(\mathbf{r}^i)] \cdot A_{(:,1)} & \dots & \mathbf{r}_M^i[\nabla_{\mathbf{r}^i}L(\mathbf{r}^i)] \cdot A_{(:,1)} \\ \mathbf{r}_1^i[\nabla_{\mathbf{r}^i}L(\mathbf{r}^i)] \cdot A_{(:,2)} & \dots & \mathbf{r}_M^i[\nabla_{\mathbf{r}^i}L(\mathbf{r}^i)] \cdot A_{(:,2)} \\ \dots & \mathbf{r}_k^i[\nabla_{\mathbf{r}^i}L(\mathbf{r}^i)] \cdot A_{(:,j)} & \dots \\ \mathbf{r}_1^i[\nabla_{\mathbf{r}^i}L(\mathbf{r}^i)] \cdot A_{(:,M)} & \dots & r_M^i[\nabla_{\mathbf{r}^i}L(\mathbf{r}^i)] \cdot A_{(:,M)} \end{bmatrix}$$

$= LHS.$

Hence

$$\nabla_w L(\mathbf{r}^i(W)) = \left(\mathbf{r}^i \left[\nabla_{\mathbf{r}^i}L(\mathbf{r}^i)\right]^T \left[I - G^iW\right]^{-1} G^i\right)^T$$

Simplifying this expression gives Eq. (5.6) for ΔW_1^i.

A.2 Analysis of A Natural Reparameterization Learning Rule

We now consider the updates given by the re-parameterization $A = \left[[G^i]^{-1} - W\right]^{-1}$. The direct re-parameterized update, ΔW_2^i in this case is given by

$$\begin{aligned}
\Delta W_2^i &= -\left[\left(A - \eta_A(\nabla_{\mathbf{r}^i} L)(\mathbf{x}^i)^T\right)^{-1} - A^{-1}\right] \\
&= -\left[\left(\left[[G^i]^{-1} - W\right]^{-1} - \eta_A(\nabla_{\mathbf{r}^i} L)(\mathbf{r}^i)^T \left[[G^i]^{-1} - W^T\right]\right)^{-1} - [G^i]^{-1} - W\right].
\end{aligned}$$

Proof: Since $A = \left[[G^i]^{-1} - W\right]^{-1}$, we have $W = [G^i]^{-1} - A^{-1}$. Let W^0 and A^0 represent the previous step update before W and A, then for each learning step iteration, $i-$iteration, $G^i = [G^i]^0$ since G^i is i dependent only and regardless of learning steps, meaning the fixed change of the coordinate.

$$\begin{aligned}
\Delta W^i &= W - W^0 \\
&= [G^i]^{-1} - A^{-1} - \left(\left[[G^i]^0\right]^{-1} - [A^0]^{-1}\right) \\
&= ([G^i]^{-1} - \left[[G^i]^0\right]^{-1}) - A^{-1} + [A^0]^{-1} \\
&= -\left(A^0 + \Delta A\right)^{-1} + [A^0]^{-1} \\
&= -\left(A^0 - \eta_A \left(\nabla_{\mathbf{r}^i} L\right) \left(\mathbf{x}^i\right)^T\right)^{-1} + [A^0]^{-1} \\
&= -\left(A^0 - \eta_A \left(\nabla_{\mathbf{r}^i} L\right) \left(\mathbf{r}^i\right)^T [A^0]^{-T}\right)^{-1} + [A^0]^{-1}.
\end{aligned}$$

To get the expression that has only G^i and W, we can substitute A and $A^{-1} = [G^i]^{-1} - W$, and since $G = G^T$ and $G^{-T} = G^{-1}$ as G is a diagonal matrix.

$$\begin{aligned}
\Delta W_2^i &= -\left(A - \eta_A \left(\nabla_{\mathbf{r}^i} L\right) \left(\mathbf{r}^i\right)^T A^{-T}\right)^{-1} + A^{-1} \\
&= -\left(\left[[G^i]^{-1} - W\right]^{-1} - \eta_A \left(\nabla_{\mathbf{r}^i} L\right) \left(\mathbf{r}^i\right)^T \left[[G^i]^{-1} - W^T\right]\right)^{-1} + \left[[G^i]^{-1} - W\right].
\end{aligned}$$

This completes the proof. Note that as $G_{jj}^i \to 0$, $A_{jj}^{-1} = [G_{jj}^i]^{-1} - W_{jj} \to \infty$, so this reparameterizatin is poorly behaved in situations where $G_{jj} = f'(\mathbf{z}_j)$ becomes small or zero. The final term will blow up.

We also show that linearizing this parameterization around $\eta_A = 0$ still leads to updates that blow up when elements of G become small. Following the linearization from Section 5.3.2, the linearized, re-parameterized update is given by

$$\begin{aligned}\Delta W_3^i &= -\eta_A A^{-1}(\nabla_{\mathbf{r}^i} L)(\mathbf{x}^i)^T A^{-1}\\ &= -\eta_A \left[[G^i]^{-1} - W\right](\nabla_{\mathbf{r}^i} L)(\mathbf{r}^i)^T \left[[G^i]^{-1} - W^T\right]\left[[G^i]^{-1} - W\right].\end{aligned}$$

Proof: For a small disk center at a, Taylor expansion of $\Delta W_2^i(\eta_A)$ is

$$\Delta W_2^i(\eta_A) = \Delta W_2^i(a) + \frac{\Delta W_2^{i\prime}(a)}{1!}(\eta_A - a) + \frac{\Delta W_2^{i\prime\prime}(a)}{2!}(\eta_A - a)^2 + \dots$$

consider a center at 0 (so $a = 0$), let $V = A + \Delta A = A - \eta_A (\nabla_{\mathbf{r}^i} L)\left(A^{-1}\mathbf{r}^i\right)^T$, then $\Delta W_2^i = A^{-1} - V^{-1}$, we can linearize it around η_A to get a new learning rule,

$$\begin{aligned}\Delta W_3^i(\eta_A) &\approx \Delta W_2^i(0) + \frac{\Delta W_2^{i\prime}(0)}{1!}(\eta_A - 0)\\ &= \left(A^{-1} - V^{-1}\right)\Big|_{\eta_A=0} + \left[\frac{dA^{-1}}{d\eta_A}\Big|_{\eta_A=0} - \left(-V^{-1}\frac{dV}{d\eta_A}V^{-1}\right)\right]\eta_A\\ &= \left(A^{-1} - \left(A^{-1} - 0\right)\right) + 0 + V^{-1}\frac{dV}{d\eta_A}V^{-1}\Big|_{\eta_A=0}\eta_A\\ &= V^{-1}\left(-\left(\nabla_{\mathbf{r}} L\right)\left(A^{-1}\mathbf{r}^i\right)^T\right)V^{-1}\Big|_{\eta_A=0}\eta_A\\ &= -\left(A^{-1} - 0\right)(\nabla_{\mathbf{r}^i} L)(A^{-1}\mathbf{r}^i)^T\left(A^{-1} - 0\right)\eta_A\\ &= -A^{-1}(\nabla_{\mathbf{r}^i} L)\left(\mathbf{r}^i\right)^T A^{-T}A^{-1}\eta_A\\ &= -\left[[G^i]^{-1} - W\right]^{-1}(\nabla_{\mathbf{r}^i} L)\left(\mathbf{r}^i\right)^T\left[[G^i]^{-1} - W^T\right]\left[[G^i]^{-1} - W\right]\eta_A.\end{aligned}$$

Substitute A_i^{-1}, we have the final expression. Notice that $\Delta W_3^i = A^{-1}A^{-T}\Delta W_1^i A^{-T}A^{-1}$. One can let $B = A^{-1}A^{-T}$ and $C = A^{-T}A^{-1}$, so B and C are symmetrical matrices. $\Delta W_3^i = B\Delta W_1^i C$.

This completes the proof. Note, again, that ΔW_3^i diverges if elements of G^i go to zero.

A.3 Linearization of the New Reparameterization Rule

Here, we derive the linearized update, ΔW_3^i, given in Eq. (5.18). This update rule is derived by expanding ΔW_2^i in Eq. (5.17) to linear order. Recall that ΔW_2^i was derived from the re-parameterization $A = [G^i - G^i W G^i]$.

Proof: Let $U = [G^{-1} - W]^{-1}$, then we can rewrite Eq. (5.17) as

$$\Delta W_2^i = -\left[U - \eta_A \left[G^i\right]^2 (\nabla_{\mathbf{r}^i} L) (\mathbf{r}^i)^T \left[I - G^i W\right]^T G^i\right]^{-1} + U^{-1}.$$

Now, denote everything inside of the inverse bracket as, V, so

$$V = U - \eta_A \left[G^i\right]^2 (\nabla_{\mathbf{r}^i} L) (\mathbf{r}^i)^T \left[I - G^i W\right]^T G^i,$$

then Eq. 5.17 can be further rewritten as

$$\Delta W_2^i = U^{-1} - V^{-1}.$$

Perform the Taylor expansion of $\Delta W_2^i(\eta_A)$ centered at $\eta_a = 0$ to first order in η_A

similar to the approach in Appendix A.2:

$$\begin{aligned}
\Delta W_3^i(\eta_A) &\approx \Delta W_2^i(0) + \frac{\Delta W_2^{i\prime}(0)}{1!}(\eta_A - 0) \\
&= \left(U^{-1} - V^{-1}\right)\Big|_{\eta_A=0} + \left[\frac{dU^{-1}}{d\eta_A}\Big|_{\eta_A=0} - (-V^{-1}\frac{dV}{d\eta_A}V^{-1})\right]\eta_A \\
&= \left(U^{-1} - \left(U^{-1} - 0\right)\right) + 0 + V^{-1}\frac{dV}{d\eta_A}V^{-1}\Big|_{\eta_A=0} \quad \eta_A \\
&= V^{-1}\left(-\left(\nabla_{\mathbf{r}}L\right)\left(A^{-1}\mathbf{r}^i\right)^T\right)V^{-1}\Big|_{\eta_A=0} \quad \eta_A \\
&= -\left(U^{-1} - 0\right)(\nabla_{\mathbf{r}^i}L)(A^{-1}\mathbf{r}^i)^T\left(A^{-1} - 0\right)\eta_A \\
&= -U^{-1}(\nabla_{\mathbf{r}^i}L)\left(\mathbf{r}^i\right)^T A^{-T}A^{-1}\eta_A \\
&= -\left[[G^i]^{-1} - W\right]^{-1}\left[G^i\right]^2(\nabla_{\mathbf{r}^i}L)\left(\mathbf{r}^i\right)^T\left[I - G^iW\right]^T G^i\left[[G^i]^{-1} - W\right]\eta_A.
\end{aligned}$$

Simplify the last line by distributing G^i into U^{-1}, we obtain Eq. 5.18

$$\Delta W_3^i = -\eta_A\left[I - WG^i\right]G^i\left(\nabla_{\mathbf{r}^i}L\right)(\mathbf{r}^i)^T\left[I - G^iW\right]^T\left[I - G^iW\right].$$

This completes the proof.

A.4 Convexity of the Cost Function for the Linear One-dimensional Model.

Here we prove that the cost function, $J(w)$ for the linear model in $N = 1$ dimension from Section 5.4.1 is convex over the stability region of w.

Proof: To see $J''(W) > 0$. When $N = 1$, X, R, and Y are vectors with dimension $1 \times m$, where m is the sample size, note that $R_i = [(I - W)^{-1}X]_i = \frac{1}{1-w}X_i$ is a scalar.

Since $J(W) = \sum_{i=1}^{m} L_i(W)$, This is equivalent to show for each sample, i$= 1, 2, \ldots, m$,

$L_i''(R_i(W)) = \frac{d^2\left[(R_i - Y_i)^2\right]}{dW^2} > 0$. Calculate the first derivative:

$$\begin{aligned}
L_i{}'(R_i(w)) &= \frac{dL}{dR_i}\frac{dR_i}{dw} \\
&= 2(R_i - Y_i)\left[\frac{-1}{(1-w)^2}X_i\right](-1) \\
&= 2(R_i - Y_i)\frac{X_i}{(1-w)^2} \\
&= \frac{2}{(1-w)^2}(R_i - Y_i)X_i.
\end{aligned}$$

Set the last expression to zero to find a minimum or maximum value. For each sample i, when firing rate reach to its target, $R_i = Y_i$, we have $L_i'(R_i(W)) = 0$. For concavity, we need to check the second derivative:

$$\begin{aligned}
L_i''(R(w)) &= 2\left[\frac{d\left(\frac{1}{(1-w)^2}\right)}{dw}(R_i - Y_i)X_i + \frac{1}{(1-w)^2}\frac{d\left(R_i - Y_i\right)X_i}{dw}\right] \\
&= 2\left[\frac{-2}{(1-w)^3}(-1)\left(\frac{X_i}{1-w} - Y_i\right)X_i + \frac{1}{(1-w)^2}\frac{-1}{(1-w)^2}X_i(-X_i)\right] \\
&= 2\left[\frac{3X_i^2}{(1-w)^4} - \frac{2Y_iX_i}{(1-w)^3}\right] \\
&= 2\frac{1}{(1-w)^4}\left[3X_i^2 - 2Y_iX_i(1-w)\right].
\end{aligned}$$

For convexity, we want $L_i''(R_i(w)) > 0$, then from $3X_i^2 - 2Y_iX_i(1-w) > 0$, we can solve for Y_i. Since $1 - w > 0$, we can arrange terms such that $\frac{3}{2}\frac{X_i}{1-w} > Y_i$. Hence, as long as the firing rates are not so much away from the targets (this is equivalent to say "if our data is linear, then convexity is guaranteed". In Example 5.4.1 linear 1D network setup, we add the noise term Eq. (5.29) to control how much "linear" in between X and Y, so $\sigma_y N(0,1)$ term needs to be within half times in comparison to generate X_i; otherwise data would look like a cluster of "cloud", hence non-linear. More specifically when $Y_i < \frac{3}{2}R_i$ for all i, we have a convex loss with respect to w.

In our 1D example, $W = -1$, $X_i \sim 0.1 * N(0,1)$ and $Y_i \sim \frac{1}{1-w}X_i + 0.01 * N(0,1)$. This is equivalent to $Y_i \approx \frac{1}{2}X_i + \frac{1}{10}X_i = 0.6X_i$, and this is indeed less than $\frac{3}{2}\frac{X_i}{1-(-1)} = 0.75X_i$ condition, so the convexity in Figure 5.2 is guaranteed.

A.5 Stability Region Boundary

Here, we derive the stability boundary on W from the model in Section 5.4.2. Given a line segment $t \in [-1,1]$ or a xy-grid with $[-1,1] \times [-1,1]$, want to find the stable initial location when performing gradient descent updates.

Proof: Let Z be a $N \times N$ random matrix with independent and identically distributed entries in the limit of $N \to \infty$ such that $Z_{ij} \sim \frac{1}{\sqrt{N}}N(0,1)$, then by circular law, the spectral radius of Z, $\rho(Z) = 1$. Furthermore, $\sigma^2(Z) = \frac{1}{N}$ and $\sigma(Z) = \frac{1}{\sqrt{N}}$. One can consider $\rho(Z)$ as the numerator part of the $\sigma(Z)$.

Since in our underline $\hat{W} = \sigma_w * Z$, then $\rho(\hat{W}) = \sigma_w * 1$. Similarly, $\sigma^2(\hat{W}) = \frac{\sigma_w^2}{N}$ and $\sigma(\hat{W}) = \frac{\sigma_w}{\sqrt{N}}$. Notice that for stability, we want $\rho(\hat{W}) < 1$, so $\sigma_w \in (0,1)$.

Now if we do some perturbation W_1 from the optimal $\hat{W}$, let $W_1 = p * \hat{W}$ along a random direction in the line segment parameter space, we have $W = t * W_1 + W^*$. The goal is to find the stability part in the segment t. For stability, we want the eigenvalue of the Jacobian matrix to be all negative, or the spectral radius $\rho(W) < 1$. Assume independence between W_1 and W^*, since $W_1 \sim p * \sigma_w Z$ and $W^* = ([G^i]^{-1}Y - X)Y^+$, where X is independently generated from $X \sim \sigma_x N(0,1)$. Hence, $\sigma^2(W) = t^2\sigma^2(W_1) + \sigma^2(W^*) \approx \frac{t^2p^2\sigma_w^2}{N} + \frac{\sigma_w^2}{N}$ since $\hat{W} \approx W^*$ in magnitude. Therefore, $\sigma(W) = \frac{\sigma_w\sqrt{t^2p^2+1}}{\sqrt{N}}$. Hence, $\rho(W) = \sigma_w\sqrt{t^2p^2+1} < 1$. Solve for t, we have $t^2 < \frac{1-\sigma_w^2}{\sigma_w^2p^2}$, equivalently $-\frac{\sqrt{1-\sigma_w^2}}{\sigma_w p} < t < \frac{\sqrt{1-\sigma_w^2}}{\sigma_w p}$. This is the boundary in Example 5.4.1.

If consider perturb $\hat{W}$ in a random orientated plane in Example 5.4.2 , we have $W = x{*}W_1 + y{*}W_2 + W^*$, where $W_l = p{*}\hat{W}$ for $l = 1, 2$. To find the stability region in the xy-plane, we have $\sigma^2(W) = x^2\sigma^2(W_1) + y^2\sigma^2(W_2) + \sigma^2(\hat{W}) = \frac{x^2p^2\sigma_w^2}{N} + \frac{y^2p^2\sigma_w^2}{N} + \frac{\sigma_w^2}{N}$ and $\sigma(W) = \frac{\sigma_w\sqrt{x^2p^2+y^2p^2+1}}{\sqrt{N}}$. Hence, $\rho(W) = \sigma_w\sqrt{x^2p^2 + y^2p^2 + 1} < 1$. Solve for the point (x,y), we have $x^2 + y^2 < \frac{1-\sigma_w^2}{\sigma_w^2p^2}$.

A.6 Optimal Parameters in Linear Networks

Here, we analyze the optimal parameters, W^*, for linear networks with $f(z) = z$ and $L() =$ in any number of dimensions, $N \geq 1$. The cost function can be written as

$$J(W) = \left([I - W]^{-1}X - Y\right)^T \left([I - W]^{-1}X - Y\right)$$

and we wish to find a minimizer, W^*, of $J(W)$. One approach to finding an optimal value is to reparameterize the system using $A = [I - W]^{-1}$ so that the cost can be re-written as

$$J_A(A) = (AX - Y)^T (AX - Y).$$

In this case, finding the optimal A is a standard least squares problem.

Proof: In the under-parameterized case ($m \geq N$) when XX^T is full rank, $A^* = YX^T[XX^T]^{-1}$ is the unique minimizer of $J(W)$ and therefore

$$W^* = I - [A^*]^{-1} = I - XX^T[YX^T]^{-1}$$

is the unique minimizer of $J(W)$ when $m \geq N$, XX^T is full rank, and YX^T is full rank. Note that when $N = 1$, XX^T and YX^T are scalars and $W^* = 1 -$

$(XX^T)/(YX^T)$.

In the over-parameterized case ($m < N$) when XX^T is full rank, there are infinitely many choices of A for which $J(A) = 0$. A common minimizer is given by

$$A^* = YX^+$$

where X^+ is the Moore-Penrose pseudo-inverse of X. This is the solution to $AX = Y$ that minimizes the Frobenius norm of A, *i.e.*,

$$A^* = \text{argmin}_A \|A\| \ \text{ s.t. } \ AX = Y$$

where $\|\cdot\|$ is the Frobenius norm. We could again find a minimizer of $J(W)$ by taking $W^* = I - [A^*]^{-1}$. However, this would represent a solution, W, that minimizes the norm of $A = [I - W]^{-1}$. Since stability is promoted by W having a small spectral radius (all eigenvalues of W must have a real part less than 1 for stability), this is a poor choice of W^*. Minimizing the Frobenius norm of A will tend to push the eigenvalues of A toward zero, which can lead to large eigenvalues of $W = I - A^{-1}$.

Instead, to find a good optimizer, W^*, in the under-parameterized case ($m \geq N$), we should find solutions that minimize the norm of W instead of A. In the under-parameterized case when XX^T is full rank, optimal W satisfies $J(W) = 0$. In other words, we wish to solve

$$W^* = \text{argmin}_W \|W\| \ \text{ s.t. } \ [I - W]^{-1} X = Y$$

To solve this problem, we re-write it as a more standard least squares problem

$$W^* = \text{argmin}_W \|W\| \ \text{ s.t. } \ WY = Y - X.$$

This problem has a solution

$$W^* = [Y - X]Y^+$$

where Y^+ is the Moore-Penrose pseudo-inverse of Y. This solution is the solution with minimum Frobenius norm and is, therefore, more likely to have a smaller spectral and therefore more likely to give stable fixed points. Hence, this is a good optimizer in the over-parameterized case ($N > m$).

A.7 Learning Rules for RNNs with Nonlinear Activation Functions.

We first prove Eq. (5.58).

Proof: Given Eq. (5.52) such that $L(\mathbf{y}^i, \mathbf{s}^i) = -\sum_{l=1}^{C} \mathbf{y}_l^i log(\mathbf{s}_l^i)$, with $\mathbf{y}^i$ represents one sample of the C-classes label output and it is coded as a "one-hot" vector, so binary entries. Soft-max function in Eq. (5.53) $\mathbf{s}_l^i = \frac{e^{\mathbf{z}_l^i}}{\sum_{k=1}^{C} e^{\mathbf{z}_k^l}}$. For each sample, i, let $\mathbf{r}^i$ be the firing rate, then $\mathbf{r}^i \in \mathcal{R}^N$, hence $Z \in \mathcal{R}^C$, where $\mathbf{z}^i = W_{out}\mathbf{r}^i$ is the $i-$th readout output from the network.

$$\begin{aligned}\Delta W_1^i(p,q) &= -\eta_W \frac{dL}{dW} \\ &= \sum_{j,k=1}^{C,C} \frac{\partial L}{\partial \mathbf{s}_j^i} \frac{\partial \mathbf{s}_j^i}{\partial \mathbf{z}_k^i} \frac{\partial \mathbf{z}_k^i}{\partial W_{pq}}.\end{aligned}$$

We start with the element-wise level first. From Appendix A.1 the Lemma 1, we know

$$\begin{aligned}\frac{\partial \mathbf{z}^i}{\partial W_{pq}} &= W_{out} \frac{\partial \mathbf{r}^i}{\partial W_{pq}} \\ &= W_{out}\left[[I - W]^{-1} \mathbf{1}_{qp} \mathbf{r}^i\right] \\ &= \mathbf{r}_q^i \left[W_{out}\left[I - W\right]^{-1}\right]_{(:,p)}\end{aligned}$$

Hence $\frac{\partial \mathbf{z}^i}{\partial W_{pq}} = \mathbf{r}_q^i \left[W_{out}\left[I - W\right]^{-1}\right]_{(:,p)} \in \mathcal{R}^C$ and $\frac{\partial \mathbf{z}_k^i}{\partial W_{pq}} = \mathbf{r}_q^i \left[W_{out}\left[I - W\right]^{-1}\right]_{(k,p)} \in R$.

Lemma 2

$$\frac{\partial \boldsymbol{s}_j^i}{\partial \boldsymbol{z}_k^i} = \boldsymbol{s}_j^i \left(\delta_{jk} - \boldsymbol{s}_k\right).$$

Since we know

$$\frac{dlog\left(\boldsymbol{s}_j^i\right)}{d\boldsymbol{s}_j^i} = \frac{1}{\boldsymbol{s}_j^i} \in R,$$

and

$$\frac{\partial \log\left(\boldsymbol{s}_j^i\right)}{\partial \boldsymbol{z}_k^i} = \frac{1}{\boldsymbol{s}_j^i} \frac{\partial \boldsymbol{s}_j^i}{\partial \boldsymbol{z}_k^i}$$

Rearrange terms, we have

$$\boldsymbol{s}_j^i \frac{\partial log\left(\boldsymbol{s}_j^i\right)}{\partial \boldsymbol{z}_k^i} = \frac{\partial \boldsymbol{s}_j^i}{\partial \boldsymbol{z}_k^i}.$$

Therefore

$$\begin{aligned}
\frac{\partial \mathbf{s}_j^i}{\partial \mathbf{z}_k^i} &= \mathbf{s}_j^i \frac{\partial log\left(\mathbf{s}_j^i\right)}{\partial \mathbf{z}_k^i} \\
&= \mathbf{s}_j^i \frac{\partial log\left(\frac{e^{\mathbf{z}_j^i}}{\sum\limits_{l=1}^{C} e^{\mathbf{z}_l^i}}\right)}{\partial \mathbf{z}_k^i} \\
&= \mathbf{s}_j^i \left(\frac{d\mathbf{z}_j^i}{d\mathbf{z}_k^i} - \frac{\partial log\left(\sum\limits_{l=1}^{C} e^{\mathbf{z}_l^i}\right)}{\partial \mathbf{z}_k^i}\right) \\
&= \mathbf{s}_j^i \left(\delta_{jk} - \frac{1}{\sum_{l=1}^{C} e^{\mathbf{z}_l^i}} \frac{\partial \sum_{l=1}^{C} e^{\mathbf{z}_l^i}}{\partial \mathbf{z}_k^i}\right) \\
&= \mathbf{s}_j^i \left(\delta_{jk} - \frac{e^{\mathbf{z}_k^i}}{\sum_{l=1}^{C} e^{\mathbf{z}_l^i}}\right) \\
&= \mathbf{s}_j^i \left(\delta_{jk} - \mathbf{s}_k^i\right) \in R
\end{aligned}$$

where $\delta_{jk} = 1$ whenever $j = k$; zero otherwise.

Now, putting it all together we have

$$\begin{aligned}\Delta W_1^i(p,q) &= -\eta_W \frac{1}{m}\sum_{j,k=1}^{C,C} \frac{\partial L_i}{\partial \mathbf{s}_j^i}\frac{\partial \mathbf{s}_j^i}{\partial \mathbf{z}_k^i}\frac{\partial \mathbf{z}_k^i}{\partial W_{pq}} \\ &= -\frac{\eta_W}{m}\sum_{j,k=1}^{C,C} \frac{1}{\mathbf{s}_j^i}\mathbf{s}_j^i\left(\delta_{jk}-\mathbf{s}_k^i\right)\mathbf{r}_q^i\left[W_{out}\left[I-W\right]^{-1}\right]_{(k,p)} \\ &= -\frac{\eta_W}{m}\sum_{j,k=1}^{C,C} \mathbf{r}_q^i\left(\delta_{jk}-\mathbf{s}_k^i\right)\left[W_{out}\left[I-W\right]^{-1}\right]_{(k,p)}\end{aligned}$$

In the matrix level, following the same proof as in Appendix A.1, if we let $B = [W_{out}]_{.,S}[I_{s,s} - W_{S,S}]$, then we have

$$LHS = \nabla_W L(\mathbf{r}^i(W), \mathbf{y}^i) = \begin{bmatrix} \frac{dL}{dW_{11}} & \frac{dL}{dW_{12}} & \cdots & \cdots & \frac{dL}{dW_{1S}} \\ \frac{dL}{dW_{21}} & \frac{dL}{dW_{22}} & \cdots & \cdots & \frac{dL}{dW_{2S}} \\ \frac{dL}{dW_{j1}} & \cdots & \frac{dL}{dW_{pq}} & \cdots & \frac{dL}{dW_{jS}} \\ \frac{dL}{dW_{S1}} & \frac{dL}{dW_{S2}} & \cdots & \cdots & \frac{dL}{dW_{SS}} \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{r}_1^i[\nabla_{\mathbf{s}^i} L(\mathbf{s}^i,\mathbf{y}^i)]\cdot B_{(:,1)} & \cdots & \mathbf{r}_S^i[\nabla_{\mathbf{s}^i} L(\mathbf{s}^i,\mathbf{y}^i)]\cdot B_{(:,1)} \\ \mathbf{r}_1^i[\nabla_{\mathbf{s}^i} L(\mathbf{s}^i,\mathbf{y}^i)]\cdot B_{(:,2)} & \cdots & \mathbf{r}_S^i[\nabla_{\mathbf{s}}^i L(\mathbf{s}^i,\mathbf{y}^i)]\cdot B_{(:,2)} \\ \cdots & \mathbf{r}_k[\nabla_{\mathbf{s}^i} L(\mathbf{s}^i,\mathbf{y}^i)]\cdot B_{(:,j)} & \cdots \\ \mathbf{r}_1^i[\nabla_{\mathbf{s}^i} L(\mathbf{s}^i,\mathbf{y}^i)]\cdot B_{(:,S)} & \cdots & \mathbf{r}_S^i[\nabla_{\mathbf{s}}^i L(\mathbf{s}^i,\mathbf{y}^i)]\cdot B_{(:,S)} \end{bmatrix},$$

$$RHS = \left(\mathbf{r}^i[\nabla_{\mathbf{s}^i} L(\mathbf{s}^i, \mathbf{y}^i)]^T [W_{out}]_{.,S}[I_{S,S} - W_{S,S}]^{-1}\right)^T = \left(\mathbf{r}^i[\nabla_{\mathbf{s}^i} L(\mathbf{s}^i, \mathbf{y}^i)]^T B\right)^T$$

$$= \left(\begin{bmatrix} \mathbf{r}_1^i \frac{\partial L(\mathbf{s}^i,\mathbf{y}^i)}{\partial \mathbf{s}_1^i} & \mathbf{r}_1^i \frac{\partial L(\mathbf{s}^i,\mathbf{y}^i)}{\partial \mathbf{s}_2^i} & \dots & \mathbf{r}_1^i \frac{\partial L(\mathbf{s}^i,\mathbf{y}^i)}{\partial \mathbf{s}_S^i} \\ \mathbf{r}_2^i \frac{\partial L(\mathbf{s}^i,\mathbf{y}^i)}{\partial \mathbf{s}_1^i} & \mathbf{r}_2^i \frac{\partial L(\mathbf{s}^i,\mathbf{y}^i)}{\partial \mathbf{s}_2^i} & \dots & \mathbf{r}_2^i \frac{\partial L(\mathbf{s}^i,\mathbf{y}^i)}{\partial \mathbf{s}_S^i} \\ \dots & \dots & \dots & \dots \\ \mathbf{r}_S^i \frac{\partial L(\mathbf{s}^i,\mathbf{y}^i)}{\partial \mathbf{s}_1^i} & \mathbf{r}_S^i \frac{\partial L(\mathbf{s}^i,\mathbf{y}^i)}{\partial \mathbf{s}_2^i} & \dots & \mathbf{r}_S^i \frac{\partial L(\mathbf{s}^i,\mathbf{y}^i)}{\partial \mathbf{s}_S^i} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} & \dots & B_{1S} \\ B_{21} & B_{22} & \dots & B_{2S} \\ \dots & \dots & \dots & \dots \\ B_{S1} & B_{S2} & \dots & B_{SS} \end{bmatrix} \right)^T$$

$$= \begin{bmatrix} \mathbf{r}_1^i[\nabla_{\mathbf{s}^i} L(\mathbf{s}^i, \mathbf{y}^i)]^T B_{(:,1)} & \mathbf{r}_1^i[\nabla_{\mathbf{s}^i} L(\mathbf{s}^i, \mathbf{y}^i)]^T B_{(:,2)} & \dots & \mathbf{r}_1^i[\nabla_{\mathbf{s}^i} L(\mathbf{s}^i, \mathbf{y}^i)]^T B_{(:,S)} \\ \mathbf{r}_2^i[\nabla_{\mathbf{s}^i} L(\mathbf{s}^i, \mathbf{y}^i)]^T B_{(:,1)} & \mathbf{r}_2^i[\nabla_{\mathbf{s}^i} L(\mathbf{s}^i, \mathbf{y}^i)]^T B_{(:,2)} & \dots & \mathbf{r}_2^i[\nabla_{\mathbf{s}^i} L(\mathbf{s}^i, \mathbf{y}^i)]^T B_{(:,S)} \\ \dots & \dots & \dots & \dots \\ \mathbf{r}_S[\nabla_{\mathbf{s}^i} L(\mathbf{s}^i, \mathbf{y}^i)]^T B_{(:,1)} & \mathbf{r}_S[\nabla_{\mathbf{s}^i} L(\mathbf{s}^i, \mathbf{y}^i)]^T B_{(:,2)} & \dots & \mathbf{r}_S[\nabla_{\mathbf{s}^i} L(\mathbf{s}^i, \mathbf{y}^i)]^T B_{(:,S)} \end{bmatrix}^T$$

$$= \begin{bmatrix} \mathbf{r}_1^i[\nabla_{\mathbf{s}^i} L(\mathbf{s}^i, \mathbf{y}^i)] \cdot B_{(:,1)} & \dots & \mathbf{r}_S^i[\nabla_{\mathbf{s}^i} L(\mathbf{s}^i, \mathbf{y}^i)] \cdot B_{(:,1)} \\ \mathbf{r}_1^i[\nabla_{\mathbf{s}^i} L(\mathbf{s}^i, \mathbf{y}^i)] \cdot B_{(:,2)} & \dots & \mathbf{r}_M^i[\nabla_{\mathbf{s}^i} L(\mathbf{s}^i, \mathbf{y}^i)] \cdot A_{(:,2)} \\ \dots & \mathbf{r}_k^i[\nabla_{\mathbf{s}^i} L(\mathbf{s}^i, \mathbf{y}^i)] \cdot B_{(:,j)} & \dots \\ \mathbf{r}_1^i[\nabla_{\mathbf{s}^i} L(\mathbf{s}^i, \mathbf{y}^i)] \cdot B_{(:,S)} & \dots & r_S^i[\nabla_{\mathbf{s}^i} L(\mathbf{s}^i, \mathbf{y}^i)] \cdot B_{(:,S)} \end{bmatrix}$$

$$= LHS.$$

Hence,

$$\Delta W_1^i = \begin{cases} -\frac{\eta_W}{m} \left[\mathbf{r}_S^i \left(\mathbf{s}^i - \mathbf{y}^i\right)^T [W_{out}]_{.,S} \left[I_{S,S} - W_{S,S}\right]^{-1} \right]^T & \text{on } S \\ 0 & \text{on } S^c \end{cases}$$

We next prove Eq. (5.59), taking a similar approach as before.

Proof: since

$$\begin{aligned}\frac{\partial \mathbf{z}^i}{\partial A_{pq}} &= \frac{dW_{out}\mathbf{r}^i}{dA_{pq}} \\ &= W_{out}\frac{dA\mathbf{x}^i}{dA_{pq}} \\ &= W_{out}\mathbf{1}_{pq}\left(\mathbf{x}^i\right)^T\end{aligned}$$

from Lemma 1, we have $\frac{\partial \mathbf{z}^i}{\partial A_{pq}} = \mathbf{x}^i_q[W_{out}]_{(:,p)}$.

$$\begin{aligned}\Delta A^i(p,q) &= -\frac{\eta_A}{m}\sum_{j,k=1}^{C,C}\frac{\partial L_i}{\partial \mathbf{s}^i_j}\frac{\partial \mathbf{s}^i_j}{\partial \mathbf{z}^i_k}\frac{\partial \mathbf{z}^i_k}{\partial A_{pq}} \\ &= -\frac{\eta_A}{m}\sum_{j,k=1}^{C,C}\mathbf{x}^i_q\left(\delta_{jk}-\mathbf{s}^i_k\right)[W_{out}]_{(k,p)}\,,\end{aligned}$$

Then it follows the same way as we derive in Appdix A.1 and Eq. (5.58) above, in the matrix form, we have $\Delta A = -\frac{\eta_A}{m}\left[\left(\mathbf{x}^i_S\right)\left(\mathbf{s}^i-\mathbf{y}^i\right)^T[W_{out}]_{\cdot,S}\right]^T$. Note that in the support set, the activation is linear, so it implies that $\mathbf{x}^i = A^{-1}\mathbf{r}^i$. Hence, we can write it in terms of $\mathbf{r}^i$. Then $\Delta A = -\frac{\eta_A}{m}\left[\left(A^{-1}\right)\mathbf{r}^i_S\left(\mathbf{s}^i-\mathbf{y}^i\right)^T[W_{out}]_{\cdot,S}\right]^T$. This completes the proof.

BIBLIOGRAPHY

1. H. Adesnik and M. Scanziani. Lateral competition for cortical space by layer-specific horizontal circuits. *Nature*, 464(7292):1155–1160, 2010.

2. Y. Ahmadian and K. D. Miller. What is the dynamical regime of cerebral cortex? *Neuron*, 109(21):3373–3391, 2021.

3. A. E. Akil, R. Rosenbaum, and K. Josić. Balanced networks under spike-time dependent plasticity. *PLoS Computational Biology*, 17(5):e1008958, 2021.

4. L. B. Almeida. A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In *Artificial neural networks: concept learning*, pages 102–111. Proceedings of IEEE, 1990.

5. S.-I. Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

6. S.-I. Amari and S. C. Douglas. Why natural gradient? In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*, volume 2, pages 1213–1216. IEEE, 1998.

7. D. J. Amit and N. Brunel. Model of global spontaneous activity and local structured activity during delay periods in the cerebral cortex. *Cerebral cortex (New York, NY: 1991)*, 7(3):237–252, 1997.

8. A. Attinger, B. Wang, and G. B. Keller. Visuomotor coupling shapes the functional development of mouse visual cortex. *Cell*, 169(7):1291–1302, 2017.

9. C. Baker, C. Ebsch, I. Lampl, and R. Rosenbaum. Correlated states in balanced neuronal networks. *Physical Review E*, 99(5):052414, 2019.

10. C. Baker, V. Zhu, and R. Rosenbaum. Nonlinear stimulus representations in neural circuits with approximate excitatory-inhibitory balance. *PLoS computational biology*, 16(9):e1008192, 2020.

11. J. Barral and A. D Reyes. Synaptic scaling rule preserves excitatory–inhibitory balance and salient neuronal network dynamics. *Nature neuroscience*, 19(12):1690–1696, 2016.

12. A. M. Bastos, W. M. Usrey, R. A. Adams, G. R. Mangun, P. Fries, and K. J. Friston. Canonical microcircuits for predictive coding. *Neuron*, 76(4):695–711, 2012.

13. A. Beck. *Introduction to nonlinear optimization: Theory, algorithms, and applications with MATLAB*. SIAM, 2014.

14. R. Bogacz. A tutorial on the free-energy framework for modelling perception and learning. *Journal of mathematical psychology*, 76:198–211, 2017.

15. L. J. Borg-Graham, C. Monier, and Y. Fregnac. Visual input evokes transient and strong shunting inhibition in visual cortical neurons. *Nature*, 393(6683):369–373, 1998.

16. R. Brette and W. Gerstner. Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *Journal of neurophysiology*, 94(5): 3637–3642, 2005.

17. N. Brunel. Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *Journal of computational neuroscience*, 8:183–208, 2000.

18. N. Brunel and V. Hakim. Fast global oscillations in networks of integrate-and-fire neurons with low firing rates. *Neural computation*, 11(7):1621–1671, 1999.

19. M. Capogna, P. E. Castillo, and A. Maffei. The ins and outs of inhibitory synaptic plasticity: Neuron types, molecular mechanisms and functional roles. *European Journal of Neuroscience*, 54(8):6882–6901, 2021.

20. P. E. Castillo, C. Q. Chiu, and R. C. Carroll. Long-term plasticity at inhibitory synapses. *Current opinion in neurobiology*, 21(2):328–338, 2011.

21. C. Chow, B. Gutkin, D. Hansel, C. Meunier, and J. Dalibard. *Methods and Models in Neurophysics: Lecture Notes of the Les Houches Summer School 2003*. Elsevier, 2004.

22. A. Clark. *Surfing uncertainty: Prediction, action, and the embodied mind*. Oxford University Press, 2015.

23. C. Curto and K. Morrison. Pattern completion in symmetric threshold-linear networks. *Neural computation*, 28(12):2825–2852, 2016.

24. C. Curto, J. Geneson, and K. Morrison. Fixed points of competitive threshold-linear networks. *Neural computation*, 31(1):94–155, 2019.

25. D. Dahmen, S. Grün, M. Diesmann, and M. Helias. Second type of criticality in the brain uncovers rich multiple-neuron dynamics. *Proceedings of the National Academy of Sciences*, 116(26):13051–13060, 2019.

26. R. Darshan, C. Van Vreeswijk, and D. Hansel. Strength of correlations in strongly recurrent neuronal networks. *Physical Review X*, 8(3):031072, 2018.

27. P. Dayan and L. F. Abbott. *Theoretical neuroscience: computational and mathematical modeling of neural systems.* MIT press, 2005.

28. A. L. Dorrn, K. Yuan, A. J. Barker, C. E. Schreiner, and R. C. Froemke. Developmental sensory experience balances cortical excitation and inhibition. *Nature*, 465(7300):932–936, 2010.

29. C. Ebsch and R. Rosenbaum. Imbalanced amplification: A mechanism of amplification and suppression from local imbalance of excitation and inhibition in cortical circuits. *PLoS computational biology*, 14(3):e1006048, 2018.

30. D. Ferster and K. D. Miller. Neural mechanisms of orientation selectivity in the visual cortex. *Annual review of neuroscience*, 23(1):441–471, 2000.

31. K. Friston. The free-energy principle: a unified brain theory? *Nature reviews neuroscience*, 11(2):127–138, 2010.

32. W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski. *Neuronal dynamics: From single neurons to networks and models of cognition.* Cambridge University Press, 2014.

33. V. L. Girko. Circular law. *Theory of Probability & Its Applications*, 29(4):694–706, 1985.

34. I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning.* MIT press, 2016.

35. R. Hahnloser and H. S. Seung. Permitted and forbidden sets in symmetric threshold-linear networks. *Advances in neural information processing systems*, 13, 2000.

36. B. Haider, A. Duque, A. R. Hasenstaub, and D. A. McCormick. Neocortical network activity in vivo is generated through a dynamic balance of excitation and inhibition. *Journal of Neuroscience*, 26(17):4535–4545, 2006.

37. B. Haider, M. Häusser, and M. Carandini. Inhibition dominates sensory responses in the awake cortex. *Nature*, 493(7430):97–100, 2013.

38. D. O. Hebb. *The organization of behavior: A neuropsychological theory.* Psychology Press, 2005.

39. M. Helias, T. Tetzlaff, and M. Diesmann. The correlation structure of local neuronal networks intrinsically results from recurrent dynamics. *PLoS computational biology*, 10(1):e1003428, 2014.

40. G. Hennequin, E. J. Agnes, and T. P. Vogels. Inhibitory plasticity: balance, control, and codependence. *Annual review of neuroscience*, 40:557–579, 2017.

41. L. Hertäg and C. Clopath. Prediction-error neurons in circuits with multiple neuron types: Formation, refinement, and functional implications. *Proceedings of the National Academy of Sciences*, 119(13):e2115699119, 2022.

42. L. Hertäg and H. Sprekeler. Learning prediction error neurons in a canonical interneuron circuit. *Elife*, 9:e57541, 2020.

43. J. Homann, S. A. Koay, K. S. Chen, D. W. Tank, and M. J. Berry. Novel stimuli evoke excess activity in the mouse primary visual cortex. *Proceedings of the National Academy of Sciences*, 119(5):e2108882119, 2022.

44. R. Jolivet, F. Schürmann, T. K. Berger, R. Naud, W. Gerstner, and A. Roth. The quantitative single-neuron modeling competition. *Biological cybernetics*, 99: 417–426, 2008.

45. R. Jordan and G. B. Keller. Opposing influence of top-down and bottom-up input on excitatory layer 2/3 neurons in mouse primary visual cortex. *Neuron*, 108(6):1194–1206, 2020.

46. G. B. Keller and T. D. Mrsic-Flogel. Predictive processing: a canonical cortical computation. *Neuron*, 100(2):424–435, 2018.

47. G. B. Keller, T. Bonhoeffer, and M. Hübener. Sensorimotor mismatch signals in primary visual cortex of the behaving mouse. *Neuron*, 74(5):809–815, 2012.

48. C. M. Kim and C. C. Chow. Learning recurrent dynamics in spiking networks. *Elife*, 7:e37124, 2018.

49. D. Kincaid, D. R. Kincaid, and E. W. Cheney. *Numerical analysis: mathematics of scientific computing*, volume 2. American Mathematical Soc., 2009.

50. G. Lajoie, K. K. Lin, and E. Shea-Brown. Chaos and reliability in balanced spiking networks with temporal drive. *Physical Review E*, 87(5):052901, 2013.

51. G. Lajoie, K. K. Lin, J.-P. Thivierge, and E. Shea-Brown. Encoding in balanced networks: Revisiting spike patterns and chaos in stimulus-driven systems. *PLoS computational biology*, 12(12):e1005258, 2016.

52. I. D. Landau, R. Egger, V. J. Dercksen, M. Oberlaender, and H. Sompolinsky. The impact of structural heterogeneity on excitation-inhibition balance in cortical networks. *Neuron*, 92(5):1106–1121, 2016.

53. M. Leinweber, D. R. Ward, J. M. Sobczak, A. Attinger, and G. B. Keller. A sensorimotor circuit in mouse cortex for visual flow predictions. *Neuron*, 95(6): 1420–1432, 2017.

54. Q. Liao and T. Poggio. Bridging the gaps between residual learning, recurrent neural networks and visual cortex. *arXiv preprint arXiv:1604.03640*, 2016.

55. R. Liao, Y. Xiong, E. Fetaya, L. Zhang, K. Yoon, X. Pitkow, R. Urtasun, and R. Zemel. Reviving and improving recurrent back-propagation. In *International Conference on Machine Learning*, pages 3082–3091. PMLR, 2018.

56. S. Lim and M. S. Goldman. Balanced cortical microcircuitry for spatial working memory based on corrective feedback control. *Journal of Neuroscience*, 34(20): 6790–6806, 2014.

57. A. Litwin-Kumar and B. Doiron. Slow dynamics and high variability in balanced cortical networks with clustered connections. *Nature neuroscience*, 15(11):1498–1505, 2012.

58. Y. Luz and M. Shamir. Balancing feed-forward excitation and inhibition via hebbian inhibitory synaptic plasticity. *PLoS computational biology*, 8(1):e1002334, 2012.

59. J. Martens. New insights and perspectives on the natural gradient method. *The Journal of Machine Learning Research*, 21(1):5776–5851, 2020.

60. G. Mongillo, D. Hansel, and C. Van Vreeswijk. Bistability and spatiotemporal irregularity in neuronal networks with nonlinear synaptic transmission. *Physical review letters*, 108(15):158101, 2012.

61. J. Neter, M. H. Kutner, C. J. Nachtsheim, W. Wasserman, et al. *Applied linear statistical models*. Irwin Chicago, 1996.

62. W. Nicola and C. Clopath. Supervised learning in spiking neural networks with force training. *Nature communications*, 8(1):2208, 2017.

63. M. Okun and I. Lampl. Instantaneous correlation of excitation and inhibition during ongoing and sensory-evoked activities. *Nature neuroscience*, 11(5):535–537, 2008.

64. Y. Ollivier, C. Tallec, and G. Charpiat. Training recurrent networks online without backtracking. *arXiv preprint arXiv:1507.07680*, 2015.

65. H. Ozeki, I. M. Finn, E. S. Schaffer, K. D. Miller, and D. Ferster. Inhibitory stabilization of the cortical network underlies visual surround suppression. *Neuron*, 62(4):578–592, 2009.

66. F. Pineda. Generalization of back propagation to recurrent and higher order neural networks. In *Neural information processing systems*, 1987.

67. R. Pyle and R. Rosenbaum. Highly connected neurons spike less frequently in balanced networks. *Physical Review E*, 93(4):040302, 2016.

68. R. Pyle and R. Rosenbaum. Spatiotemporal dynamics and reliable computations in recurrent spiking neural networks. *Physical review letters*, 118(1):018103, 2017.

69. R. P. Rao and D. H. Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79–87, 1999.

70. R. P. Rao and T. J. Sejnowski. 16 predictive coding, cortical feedback, and spike-timing dependent plasticity. *Probabilistic models of the brain*, page 297, 2002.

71. A. Renart, J. De La Rocha, P. Bartho, L. Hollender, N. Parga, A. Reyes, and K. D. Harris. The asynchronous state in cortical circuits. *science*, 327(5965): 587–590, 2010.

72. R. Rosenbaum and B. Doiron. Balanced networks of spiking neurons with spatially dependent recurrent connections. *Physical Review X*, 4(2):021039, 2014.

73. R. Rosenbaum, M. A. Smith, A. Kohn, J. E. Rubin, and B. Doiron. The spatial structure of correlated neuronal variability. *Nature neuroscience*, 20(1):107–114, 2017.

74. D. B. Rubin, S. D. Van Hooser, and K. D. Miller. The stabilized supralinear network: a unifying circuit motif underlying multi-input integration in sensory cortex. *Neuron*, 85(2):402–417, 2015.

75. S. Saxena and J. P. Cunningham. Towards the neural population doctrine. *Current opinion in neurobiology*, 55:103–111, 2019.

76. A. Schulz, C. Miehl, M. J. Berry II, and J. Gjorgjieva. The generation of cortical novelty responses through inhibitory plasticity. *Elife*, 10:e65309, 2021.

77. M. Spivak. *Calculus on manifolds: a modern approach to classical theorems of advanced calculus*. CRC press, 2018.

78. S. C. Surace, J.-P. Pfister, W. Gerstner, and J. Brea. On the choice of metric in gradient-based theories of brain function. *PLoS computational biology*, 16(4): e1007640, 2020.

79. D. Sussillo. Neural circuits as computational dynamical systems. *Current opinion in neurobiology*, 25:156–163, 2014.

80. D. Sussillo and L. F. Abbott. Generating coherent patterns of activity from chaotic neural networks. *Neuron*, 63(4):544–557, 2009.

81. G. Turrigiano. Too many cooks? intrinsic and synaptic homeostatic mechanisms in cortical circuit refinement. *Annual review of neuroscience*, 34:89–103, 2011.

82. C. Van Vreeswijk and H. Sompolinsky. Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science*, 274(5293):1724–1726, 1996.

83. C. van Vreeswijk and H. Sompolinsky. Chaotic balanced state in a model of cortical circuits. *Neural computation*, 10(6):1321–1371, 1998.

84. T. P. Vogels and L. Abbott. Gating multiple signals through detailed balance of excitation and inhibition in spiking networks. *Nature neuroscience*, 12(4): 483–491, 2009.

85. T. P. Vogels, H. Sprekeler, F. Zenke, C. Clopath, and W. Gerstner. Inhibitory plasticity balances excitation and inhibition in sensory pathways and memory networks. *Science*, 334(6062):1569–1573, 2011.

86. T. P. Vogels, R. C. Froemke, N. Doyon, M. Gilson, J. S. Haas, R. Liu, A. Maffei, P. Miller, C. Wierenga, M. A. Woodin, et al. Inhibitory synaptic plasticity: spike timing-dependence and putative network function. *Frontiers in neural circuits*, 7:119, 2013.

87. H. Von Helmholtz. *Handbuch der physiologischen Optik*, volume 9. Voss, 1867.

88. C. Wacongne, J.-P. Changeux, and S. Dehaene. A neuronal model of predictive coding accounting for the mismatch negativity. *Journal of Neuroscience*, 32(11): 3665–3678, 2012.

89. S. Walczak. Artificial neural networks. In *Advanced methodologies and technologies in artificial intelligence, computer simulation, and human-computer interaction*, pages 40–53. IGI global, 2019.

90. J. C. Whittington and R. Bogacz. Theories of error back-propagation in the brain. *Trends in cognitive sciences*, 23(3):235–250, 2019.

91. B. Widrow and M. E. Hoff. Adaptive switching circuits. Technical report, Stanford Univ Ca Stanford Electronics Labs, 1960.

92. R. J. Williams and J. Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural computation*, 2(4):490–501, 1990.

93. K. Wimmer, A. Compte, A. Roxin, D. Peixoto, A. Renart, and J. De La Rocha. Sensory integration dynamics in a hierarchical network explains choice probabilities in cortical area mt. *Nature communications*, 6(1):6177, 2015.

94. X. Xie, R. H. Hahnloser, and H. S. Seung. Selectively grouping neurons in recurrent networks of lateral inhibition. *Neural computation*, 14(11):2627–2646, 2002.

95. M. Xue, B. V. Atallah, and M. Scanziani. Equalizing excitation–inhibition ratios across visual cortical neurons. *Nature*, 511(7511):596–600, 2014.

96. V. Zhu and R. Rosenbaum. Evaluating the extent to which homeostatic plasticity learns to compute prediction errors in unstructured neuronal networks. *Journal of Computational Neuroscience*, 50(3):357–373, 2022.

This document was prepared & typeset with pdfLaTeX*, and formatted with* NDdiss2$_\varepsilon$ *classfile (v3.2017.2[2017/05/09])*