Hierarchical linear-nonlinear cascade models of dendritic integration

DJ Strouse

Computational and Biological Learning Laboratory
Department of Engineering
Churchill College
University of Cambridge

This dissertation is submitted for the degree of Master of Philosophy.

For creating a stimulating and fun place to work and learn, I would like to thank the faculty and students at CBL. In particular, I would like to thank Cristina Savin for making me a more critical scientist and drinker of wine, Rich Turner for making me a more careful scientist, Peter Orbanz for his generosity and patience in sharing his mathematical wisdom, and Emil Hewage, Ferenc Huszár, Diana Burk, Koa Heaukulani, and Dan Roy for the banter.

I would also like to thank Peter Dayan, Peter Latham, Maneesh Sahani, and the rest of the Gatsby for providing a "home away from home" in London, Maneesh Sahani and Zoubin Ghahramani for serving as my examiners, Peter Patrikis and the Churchill Foundation for their support, financial and otherwise, and Maneesh Sahani, Jonathan Pillow, Peter Dayan, Peter Latham, Yee Whye Teh, Mark Goldman, Matthias Bethge, Jeremy Freeman, and Eero Simoncelli for useful discussions.

Most importantly, I have had the great fortune of working on interesting problems with individuals who are both talented collaborators and good friends. I would like to thank Tiago Branco for introducing me to the messy world of experimental neuroscience and Judit Makara and Michael Häusser for having the patience to talk biology with a physicist. Finally, I would like to thank my advisor Máté Lengyel for his generosity, for providing an endless supply of ideas and criticism, and for making me a more critical scientist, Balázs Ujfalussy for always providing a sounding board for new ideas, for cheerfully weathering our comedy of errors, and for performing all of the compartmental model simulations, and to both of them for keeping science fun.

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated in the text.

Abstract

Accumulating experimental evidence suggests that nonlinear dendritic processing plays a crucial role in single-neuron information processing. Yet, while experimental methods have progressed to the point of enabling precise spatial and temporal stimulation of the dendritic tree, there currently exists no simple, canonical model for dendritic computation. Borrowing inspiration from the recent success and popularity of linear-nonlinear-Poisson (LNP) and generalized linear models (GLMs) for sensory neural coding, we here develop a hierarchical linear-nonlinear (hLN) cascade model for dendritic integration which is analytically tractable, interpretable, and can be fit to arbitrarily complex dendritic trees.

Contents

Chapter 1

Introduction

How single cells process information is a central question in computational neuroscience. Most computational models for single cells (e.g. integrate-and-fire [Izhikevich 2003], spike response models [Gerstner and Kistler 2002]) implicitly ignore the intricate geometry of a neuron's dendritic tree, treating them as "point neurons." Typically, this is done by assuming that synaptic inputs are linearly summed and passed through a single global nonlinearity (e.g. a spike-generating mechanism). By doing so, the locations of synapses are ignored, and it is assumed that any local processing in the dendritic tree (i.e. nonlinear interactions between subsets of synaptic inputs) is insignificant [Gerstner and Kistler 2002, Dayan and Abbott 2001]. From this viewpoint, it is the single cell that is the fundamental computational unit of the brain [Zador 2000].

However, accumulating experimental evidence over the last two decades speaks to the contrary. Spurred by methodological developments granting researchers precise spatial and temporal control over the inputs to single cells, including glutamate uncaging and focal stimulation, a series of experiments have demonstrated both spatial [Magee 2000, Schiller et al 2000, Polsky et al 2004, London and Häusser 2005, Losonczy and Magee 2006, Spruston 2008] and temporal [Branco et al 2010] nonlinearities in dendritic processing. Together, these results suggest that local dendritic processing is not insignificant and that the dendritic "subunit" [Poirazi et al 2003a, Poirazi et al 2003b, Polsky et al 2004], not single cells, may be the fundamental computational unit of the brain [Branco et al 2010].

Despite these experimental advances, there still exists no simple, canonical formalization of dendritic computation. At one extreme, multi-compartmental models retain as much biophysical detail as possible, modeling a cell down to the mechanics of membrane channels and propagation of electrical signals. While such models are expressive enough to exhibit the spatial and temporal dendritic nonlinearities discovered in experiments, they are both difficult to fit to experimental data [Huys et al 2006] and to analyze mathematically. Moreover, by focusing on biophysics, rather than the functional mapping from inputs to outputs, they are difficult to interpret from a computational perspective. At the opposite extreme, heuristic "two-layer network" models [Poirazi et al 2003a, Poirazi et al 2003b, Polsky et al 2004], which assume that the somatic membrane potential is produced by passing the instantaneous synaptic inputs through a two-layer linear-nonlinear cascade, are easy to analyze mathematically and interpret computationally. However, these models were designed to work only with static inputs and outputs, restricting their experimental application to artificial stimulus protocols involving brief, intense stimulation, rather than extended spike trains with realistic statistical properties. Moreover, because of the restriction to static inputs and outputs, the metrics for judging nonlinear dendritic behavior were based only on either (1) instantaneous firing rates or (2) peaks or means of somatic membrane potential, rather than predictiveness of dynamically changing firing rates or full membrane potential traces.

We propose here a model class which retains the simplicity and interpretability of the two-layer network but which can also (1) handle dynamic inputs, (2) model dendritic trees of arbitrary complexity (using a hierarchical linear-nonlinear cascade), (3) be efficiently fit directly to existing experimental

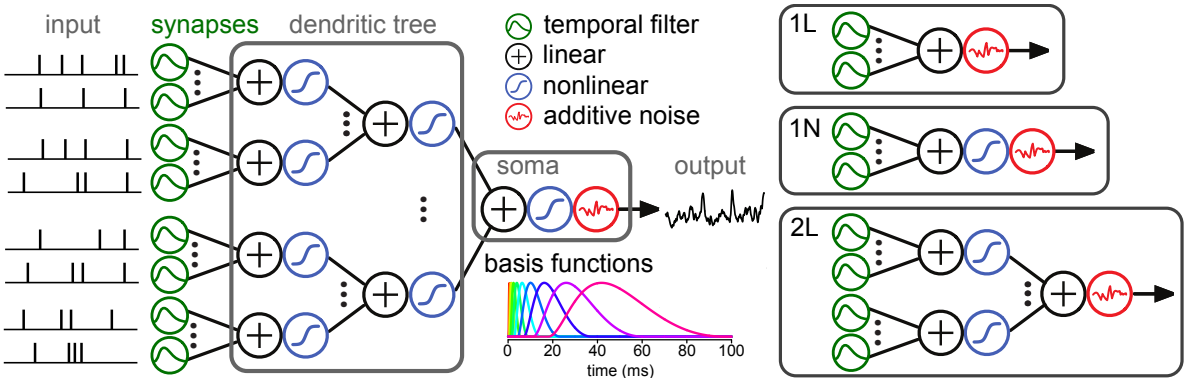

Figure 1.1: **Pictoral representation of the hLN model.** Spike train inputs are passed through temporal filters (representing synapses), followed by a hierarchy of LN cascades (representing dendrites), and a final linear or LN step (representing the soma), along with the addition of additive white Gaussian noise, to produce a somatic membrane potential. Example models at far right are a one-layer linear model (1L), one-layer nonlinear model (1N), and a two-layer linear model (2L); the larger example model at left is a three-layer nonlinear (3N) model. Basis functions at the bottom were produced according to eqn 2.4 with $M = 10$, $a = 3.75$, and $c = .01$. The horizontal axis shows time flowing into the past. Note the increased temporal resolution for more recent times. Note also that, for simplicity, the tree (hierarchy) is depicted as uniformly deep and that synapses are depicted as being connected to only the deepest subunits, but that in general, the tree may be of nonuniform depth and that synapses may be connected to any subunit (including the root subunit representing the soma). *Figure credit: Balázs Ujfalussy.*

data, and (4) be judged by its predictiveness of full somatic membrane potential traces. More specifically, the hLN model is aimed at experiments in which a number of synapses are stimulated in an arbitrarily complex spatiotemporal pattern while the somatic membrane potential of the cell is simultaneously recorded, and it is this mapping from input spike trains to membrane potential trace that we seek to learn (figure 1.1). Importantly, this model allows us to identify the appropriate computational subunits of neurons in a statistically principled, data-driven way using more natural input spike trains.

In the next chapter, we introduce the hierarchical linear-nonlinear cascade model for dendritic computation and describe how to fit it to experimental data. We then demonstrate the success of our fitting procedure using synthetic data for which the ground-truth parameters are known. Next, in order to gain confidence in our ability to make conclusive statements about nonlinear dendritic processing in real cells, we demonstrate the model's ability to distinguish data generated from linear, nonlinear, and two-layer networks though model selection. Having verified these basic properties of the hLN model, we next demonstrate its biological relevance by fitting data generated from biophysically realistic multi-compartmental models. Through a series of compartmental model experiments, we demonstrate how the nonlinear dendritic properties discovered depend on the stimulation protocol and synapse locations. Finally, we close by highlighting shortcomings of the current version of the model and directions for future work.

Chapter 2

Hierarchical linear-nonlinear cascade models

2.1 Background

Linear-nonlinear (LN) cascade models have gained popularity in the neural coding literature over the last decade as models for the spiking responses of sensory-driven neurons [Schwartz et al 2006, Pillow 2007, Pillow et al 2008]. In these cases, since the goal is to predict spikes, the LN cascade produces a firing rate which is then fed into a Poisson spike generation process, yielding a linear-nonlinear-Poisson (LNP) model. The LNP model is a special case of a more general class of models called generalized linear models (GLMs), which includes any probabilistic model with a random variable output distributed according to an exponential family distribution whose mean is produced by passing a linear combination of its inputs through a global nonlinearity [Truccolo et al 2005, Pillow 2007, Pillow et al 2008]. More recently, hierarchical LN (hLN) cascade models have been proposed as models for the multiple layers of processing that occur in sensory circuits [Freeman et al 2012, Vintch et al 2012].

Inspired by the success of GLMs and seeking to leverage a growing body of theoretical and empirical results regarding them, we propose that hLN models may be used to model dendritic processing as well. There is however an important difference between our use of these models and their typical use in the neural coding literature. In that literature, the primary model inputs are (continuous) sensory signals (such as pixel intensities) while the model outputs are (binary) spike trains. In our own application, this pattern is reversed - the model inputs are (binary) spike trains (from presynaptic cells) and the model outputs are the (continuous) somatic membrane potential of the cell. For this reason, we replace the final Poisson spike generation stage with an additive white Gaussian noise model and include a temporal filter for each input line to model the excitatory postsynaptic potentials (EPSPs).[1] In doing so, we specify a probabilistic model which can then be fit to experimental data using standard methods from statistics and machine learning. In the present case, we fit the model using maximum likelihood via coordinate ascent. A pictoral representation of the model is included in figure 1.1. We now describe the hLN model and our fitting procedure more formally.

[1] This comparison is a slight oversimplification - when feedback or cross-coupling filters are included, the GLM does also take binary spike train inputs. Indeed, it is from that application that we borrow tricks for handling the binary inputs to the hLN model.

2.2 Formal definition of the hLN model for dendritic computation

We will denote[2] the input spike train to a cell by the (binary) $T \times N_{\text{syn}}$ matrix $\mathbf{X}$, where $T\Delta t$ is the length of the experiment, time has been discretized into bins of size Δt, and N_{syn} is the number of synapses stimulated. We use a value of $\Delta t = 1$ ms since this is a typical timescale of variation in spiking and somatic membrane potential. We assume that the response of the cell at time t depends only on its inputs over a recent, finite temporal window $[t - \tau, t]$ of length τ; we use a value of $\tau = 150$ms since typical EPSP decay times are shorter. For each synapse i,[3] we then define the $T \times \tau$ (binary) synaptic input matrix $\mathbf{X}_i$ such that row t of $\mathbf{X}_i$ (denoted by $\mathbf{x}_i(t)$) constitutes the most recent τ ms of spike inputs to synapse i as of time t. The response of LN subunit j at time t in our hierarchical model is then defined as:

$$r_j(t) = g_j\left(\sum_{k \in c_{\text{sub}}(j)} w_k r_k(t) + \sum_{i \in c_{\text{syn}}(j)} \mathbf{k}_i^T \mathbf{x}_i(t)\,;\boldsymbol{\theta}_j\right), \tag{2.1}$$

where g_j is the nonlinearity for subunit j (parameterized by the vector $\boldsymbol{\theta}_j$), $c_{\text{sub}}(j)$ denotes the set of indices for the "children" of subunit j (that is, the set of subunits whose outputs are among the inputs to subunit j), $c_{\text{syn}}(j)$ denotes the set of indices for the synapses connected to subunit j, w_k denotes the weight on the output of subunit k, and $\mathbf{k}_i$ is a τ-length vector representing the temporal filter, or time-inverted EPSP, of synapse i. The inputs to subunit j thus consist of two separate terms - a weighted summation of other subunit responses (the first term in eqn 2.1) and linearly filtered versions of the subunit's recent synaptic inputs (the second term in eqn 2.1). In more biological parlance, the first term represents the influence of subbranchlet stimulation on dendritic branchlet j and the second term represents the influence of EPSPs for synapses on that branchlet. Note that the apparent recursiveness of eqn 2.1 (i.e. that subunit responses appear on both sides of the equation) is only illusory since the hierarchy on subunits (implicitly defined by $c_{\text{sub}}(j)$) imposes that the inputs to a subunit will not depend on its own output. In other words, the subunits are arranged in a (directed) tree (figure 1.1).

Letting the "root" subunit in the hierarchy be indexed by the subscript $j = 1$, we then model the somatic membrane potential at time t as:

$$\begin{aligned} \hat{v}(t) =& w_1 r_1(t) + v_0 + \eta(t) && (2.2)\\ =& \tilde{v}(t) + \eta(t) && (2.3) \end{aligned}$$

where v_0 is the baseline membrane potential (or offset, or bias), $\eta(t) \sim \mathcal{N}(0, \sigma^2)$ represents additive white Gaussian noise (of mean 0 and variance σ^2), $\tilde{v}(t) \equiv w_1 r_1(t) + v_0$ is the noiseless predicted membrane potential, and w_1 is the output gain which defines the scale of variations in the predicted somatic membrane potential.[4]

Throughout the work discussed here, we use a sigmoid (or logistic function) for the nonlinearity for all internal (i.e. non-root) subunits, defined by $g(x; x_0) = \frac{1}{1+e^{-(x+x_0)}}$, which takes an argument x and is parameterized by an input bias x_0. For the root subunit, we use either this sigmoid or the identity function $g(x) = x$ and henceforth refer to those models as having either nonlinear (N) or linear (L)

[2] Throughout this document, we will use bold uppercase letters such as $\mathbf{X}$ to denote matrices, bold lowercase letters such as $\mathbf{x}$ to denote vectors, and $\mathbf{x}^T$ to denote the transpose of the vector $\mathbf{x}$.

[3] We will use the convention that i indexes synapses (meant to connote "inputs"), j and k index LN subunits, and m indexes basis functions (to be introduced below).

[4] We use the standard conventions that $\sim$ means "distributed according to", $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes a multivariate Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, and $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the value of that distribution evaluated at $\mathbf{x}$.

output, respectively. While typically a sigmoid would also be parameterized by an output offset, an input gain, and an output gain, we do not include them explicitly, since they would be redundant with the input bias of the parent subunit (for subunits $j \neq 1$) or v_0 (for the root subunit $j = 1$), a global scale factor on the input subunit weights w_k and synaptic kernels $\mathbf{k}_i$, and the subunit output weight w_j, respectively. We chose to use a sigmoid for two reasons. First, the sigmoid has been proposed elsewhere as an appropriate dendritic nonlinearity [Poirazi et al 2003a, Poirazi et al 2003b, Polsky et al 2004]. Second, under different parameter settings and input statistics, the sigmoid is flexible enough to capture purely linear, sublinear, and superlinear behavior, as well as combinations thereof.

It is worth noting that we assume in the work discussed here that the hLN architecture is given in advance, that is that one knows $c_{\text{sub}}(j)$ and $c_{\text{syn}}(j)$ for all subunits. When fitting experimental data involving the stimulation of a small portion of the dendritic tree, one often can narrow down the possible appropriate architectures to a handful. By fitting an hLN model with each of these architectures and comparing their performances, it is possible to identify the computational subunits of a cell. In chapter 5, we perform this analysis on experimental data produced from a multi-compartmental model of a cortical pyramidal neuron. In section 6.1.1, we discuss the possibility of extending the hLN model to learning the appropriate hiearchical model structure in a more principled and automatic way.

In order to (1) reduce the number of parameters that we must infer and (2) impose temporal smoothness on the synaptic kernels, we represent $\mathbf{k}_i$ in terms of a set of M basis functions (following [Pillow et al 2008]) of the form:

$$f_m(t') = \begin{cases} \frac{1}{2}\cos(a\log[t'+c] - \phi_m) + \frac{1}{2} & t' \text{ s.t. } a\log[t'+c] \in [\phi_m - \pi, \phi_m + \pi] \\ 0 & \text{otherwise} \end{cases}, \tag{2.4}$$

where t' represents time flowing into the past and $\theta_m = \frac{m}{M}$. This family of basis functions is plotted in the bottom of figure 1.1. In words, this family constitutes a series of cosine "bumps" with varying phases in log-transformed time. We chose this set due to the property, suggested in [Pillow et al 2008], that it allows for fine-grained temporal sensitivities for recent inputs while imposing more coarse-grained temporal sensitivities for less recent inputs. Throughout the work discussed here, we use $M = 10$, $a = 3.75$, and $c = .01$. These numbers were simply chosen to yield realistic timescales for the temporal sensitivities of synapses.

By convolving the rows of the synaptic input matrices $\mathbf{X}_i$ with this set of basis functions to create the $T \times M$ convolved synaptic input (or design) matrices $\tilde{\mathbf{X}}_i$ (with rows $\tilde{\mathbf{x}}_i(t)$), we can rewrite eqn 2.1 as:

$$r_j(t) = g_j\left(\sum_{k \in c_{\text{sub}}(j)} w_k r_k(t) + \sum_{i \in c_{\text{syn}}(j)} \mathbf{b}_i^T \tilde{\mathbf{x}}_i(t)\,; \boldsymbol{\theta}_j\right), \tag{2.5}$$

where the only differences are that $\mathbf{x}_i(t)$ has been replaced by $\tilde{\mathbf{x}}_i(t)$ and the synaptic filter $\mathbf{k}_i$ has been replaced with its basis weight representation denoted by the M-length vector $\mathbf{b}_i$. Note that this change reduces the number of synaptic parameters we need to estimate from $N_{\text{syn}}\tau$ to $N_{\text{syn}}M$. For the values of these parameters we consider ($N_{\text{syn}} = 10 - 40$, $\tau = 150$, and $M = 10$), this constitutes a 15-fold reduction from $1500 - 6000$ to $100 - 400$ parameters.

2.3 Parameter estimation: overview

Our goal is to fit this model to experimental data in the form of the synaptic inputs $\mathbf{X}_i$ and simultaneously recorded somatic membrane potential $\mathbf{v}$ (a T-length vector whose t^{th} entry is the membrane potential at time t). A natural method for doing so is via maximum likelihood (ML), that is by maximizing the conditional probability of $\mathbf{v}$ given the inputs $\mathbf{X}_i$ and model parameters. Due to our

assumption of additive white Gaussian noise, ML fitting is equivalent to choosing the parameters which minimize the squared training error (commonly referred to as the "least squared error", or LSE, parameters). Concatenating the temporal history of our noiseless predicted membrane potential into the T-length vector $\tilde{\mathbf{v}}$, the N_{syn} design matrices $\tilde{\mathbf{X}}_i$ into a single $T \times (N_{\text{syn}}M)$ design matrix $\tilde{\mathbf{X}}$, the subunit output weights into an N_{sub}-length vector $\mathbf{w}$, the N_{syn} synaptic basis weight vectors $\mathbf{b}_i$ into a single $N_{\text{syn}}M$-length vector $\mathbf{b}$, and the N_{sub} nonlinearity parameter vectors $\boldsymbol{\theta}_j$ into a single parameter vector $\boldsymbol{\theta}$, we can write the log likelihood of the model parameters $\mathbf{w}$, $\mathbf{b}$, $\boldsymbol{\theta}$, v_0, and σ^2 as:

$$l\left(\mathbf{w}, \mathbf{b}, \boldsymbol{\theta}, v_0, \sigma^2\right) = \log P\left(\mathbf{v} \mid \tilde{\mathbf{X}}, \mathbf{w}, \mathbf{b}, \boldsymbol{\theta}, v_0, \sigma^2\right) \tag{2.6}$$

$$= \log \mathcal{N}\left(\mathbf{v}; \tilde{\mathbf{v}}\left(\tilde{\mathbf{X}}, \mathbf{w}, \mathbf{b}, \boldsymbol{\theta}, v_0\right), \sigma^2 \mathbf{1}\right). \tag{2.7}$$

Unfortunately, unlike the ML problem for the LNP model used in neural coding [Paninski 2004], our ML problem is not convex, potentially making optimization more difficult, or even impractical, for our model. Fortunately, despite the nonconvexity, we found that maximization of eqn 2.7 via subunit-wise coordinate ascent performed well (figures 3.2-3.3).

Intuitively, we begin by randomly initializing the model parameters $\mathbf{w}$, $\mathbf{b}$, $\boldsymbol{\theta}$, v_0, and σ^2 with "reasonable" values, then iterate over subunits j, maximizing eqn 2.7 with respect to w_j, $\mathbf{b}_j$, $\boldsymbol{\theta}_j$, v_0, and σ^2 while keeping w_k, $\mathbf{b}_k$, and $\boldsymbol{\theta}_k$ fixed for $k \neq j$ until convergence.

2.4 Parameter estimation: initialization

For "reasonable" initial parameters, we choose ones such that the subunit sigmoids are not saturated during the entire experiment. It is important to avoid that parameter regime because the derivatives of the log likelihood would then be very small, leading to very slow performance for coordinate ascent. More specifically, we initialize $\mathbf{w}$, $\mathbf{b}$, and $\boldsymbol{\theta}$ and then rescale them such that for each sigmoidal subunit, the input mean and rms deviation about the mean are -1.5 and 2.5, respectively, which empirically corresponds to the "active", unsaturated input regime. If the root (output) subunit ($j = 1$) was linear, we fixed $w_1 = 1$, since it is redundant with the w_k and $\mathbf{b}_i$ for $k \in c_{\text{sub}}(1)$ and $i \in c_{\text{syn}}(1)$, and rescaled w_k and $\mathbf{b}_i$ so that the range (i.e. difference between maximum and minimum) of the predicted membrane potential was 15mV.

Each element of the nonlinearity parameter vector $\boldsymbol{\theta}$ was initialized to -6, though its initial value was not important since it was rescaled as described above anyways. For the non-root, or "internal", subunits ($j \neq 1$), the subunit output weight w_j was drawn uniformly from the interval $[2, 3]$, while for the root subunit ($j = 1$), w_1 was drawn from the interval $[13, 17]$. Finally, the basis weights $\mathbf{b}_i$ for each synapse i were chosen to be sparse and such that the kernel $\mathbf{k}_i$ was unimodal. Unimodality was encouraged by selecting consecutive nonzero elements from 1:M, that is basis elements with similar maxima (see eqn 2.4), and choosing their values so that one "primary" element was largest and elements of increasing distance from the primary element had decreasing magnitudes. More specifically, we first choose the number of nonzero elements of $\mathbf{b}_i$ from the integer interval $[1, 3]$, select one element $b_i^{(p)}$ at random as the "primary" element, draw that element from $\mathcal{N}(10, 2)$, draw the remaining nonzeros elements $b_i^{(m)}$ from $\mathcal{N}(10\alpha, 2\alpha)$ where $\alpha = \frac{1}{2}^{|m-p|}$, and reset any nonzero elements with a value less than 1 to have value 1. Essentially, this seemingly complex procedure is really just an ad-hoc method for avoiding parameter regimes with saturated sigmoids and therefore tiny derivatives of the log likelihood.

2.5 Parameter estimation: optimization details

For the coordinate optimization iteration order, we typically begin with the subunits furthest from the root (randomizing the order within a layer) and sweep through the tree towards the root, cycling over all subunits several times until converence. The motivation for this ordering is that we optimize

the parameters of subunit j only after optimizing the parameters of all subunits $k \neq j$ whose outputs constitute the inputs to subunit j.

Maximization of eqn 2.7 for each iteration is performed via Matlab's fmincon function with non-negativity constraints on $\mathbf{w}$ and σ and a non-positivity constraint on v_0, using the trust-region-reflective algorithm - "a subspace trust-region method... based on the interior-reflective Newton method" (Matlab documentation). We placed the non-negativity constraint on $\mathbf{w}$ to eliminate a symmetry in the model which involves flipping the sign of $\mathbf{w}$ and $\mathbf{b}$ and adjusting $\boldsymbol{\theta}$ so that the input-output function of the model is unchanged but the sigmoids are activated "backwards" – that is, the sigmoids have high activation at baseline, receive negative activation during synaptic stimulation, but due to a negative output weight, the end result is still the same as normal. We used the trust-region-reflective algorithm because it was the recommended algorithm (Matlab documentation) for cases in which it was possible to provide analytic derivatives (we could) and only bound or equality constraints were needed but not both (we had only bounds).

We defined convergence of the coordinate optimization procedure using one of two conditions, whichever was met first: (1) two consecutive interations during which the log likelihood increased by less than 1[5] or (2) the completion of 20 iterations.

For coordinate optimization of a linear output subunit, it is possible to obtain closed-form solutions for the ML parameters. Thus for those optimization steps, we simply updated the parameters accordingly.

We chose to use coordinate optimization, rather than optimizing all parameters at once, since (1) we avoid calculating the cross-subunit terms in the Hessian, (2) the Hessian we must store in memory is orders of magnitude smaller, and (3) the individual coordinate optimization problems can be solved reasonably rapidly. Moreover, coordinate optimization approaches (e.g. expectation-maximization, or EM) have been known to work well in many contexts. Most importantly, as we demonstrate in the next chapter, this method works in practice in our own context.

[5] In retrospect, this is not a good choice of convergence condition since the meaning of the log likelihood changes with the amount of data. Thus a more appropriate convergence condition would depend on the length of the experiment.

Chapter 3

Validation of fitting

Before applying the hLN model to experimental data sets, we validated our fitting procedure on synthetic data sets produced from the hLN model. The purpose of doing so was to test our fitting procedure in a setting in which (1) we knew the model was appropriate and (2) we knew the underlying parameters of the model which generated the data and so could compare our inferred parameters with the true ones. In this way, we could separate validation of the fitting procedure (here) from validation of the model's biological relevance (see chapter 5).

More specifically, we generated data from the following models: (1) a single linear subunit (1L) with 10, 15, or 20 synapses, (2) a 2-layer, 3-subunit model with linear output (2L) and 10, 15, or 20 total synapses divided between the two leaf subunits as 5 and 5, 5 and 10, or 10 and 10 synapses, respectively, (3) a 3-layer, 7-subunit model with linear output (3L) and 20, 25, 30, 35, or 40 total synapses divided between the four leaf subunits as 5 each, one with 10 and three with 5, two each with 10 and 5, three with 10 and one with 5, and 10 each, respectively, and (4-6) the same models with output nonlinearities (1N, 2N, and 3N). The 1L, 1N, 2L, and 3N models are depicted in figure 1.1. The motivation for varying the number of synapses was to test how the parameter estimation performance and speed would scale to data sets of experimentally interesting sizes.

For the stimulation protocol, each synapse received an independent Poisson spike train with 10Hz firing rate.

For each model, we generated 25 instantiations with different randomized parameters, generated according to the procedure for optimization initialization described in section 2.4,[1] for each of four increasing training data set sizes (1s, 2s, 4s, and 8s) and ran the estimation procedure described in sections 2.3-2.5. The motivation for varying the training data set size was to test how much data might be required in later experiments using compartmental models or real cells.

The inferred models were then tested on a 4s test data set produced by the same model which produced the training data. We kept the size of the test data set constant (i.e. independent of the training data set size) because test performance variability would increase for smaller test data set sizes and we did not want to unfairly bias our test metrics against models training on smaller data sets. We measured performance both by the predictive power on the test data set (figure 3.2), as well as by direct comparison of the inferred to true model parameters (3.3). We judged performance based on the test set rather the training set in order to convince ourselves that "good" performance was not just due to overfitting. This is especially important when doing model selection (chapter 4), since a more complicated model can always achieve better training performance than a simpler one. The experimental protocol is summarized in figure 3.1.

The results are shown in figures 3.2 and 3.3. For the one-layer models, predictive performance of the inferred model on the test data set approached that of the true model (i.e. the level limited by noise) by the time the training data sets reached 2-4s. The parameter estimation performance

[1] The procedure was actually first developed for generating synthetic data sets and then borrowed for optimization initialization.

number of synapses	10-40
input firing rate	10Hz
correlations?	no
size of training set	1-8s
size of test set	4s
trials per condition	25

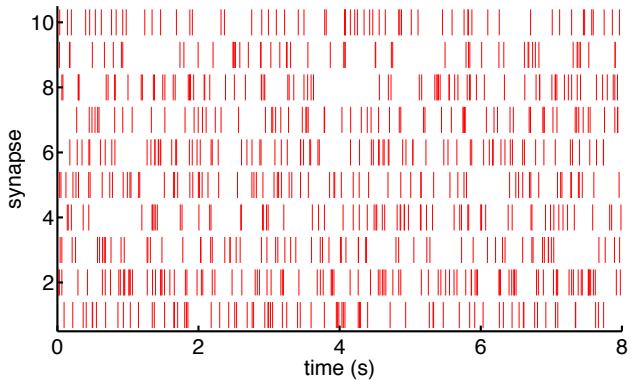

Figure 3.1: **Summary of experimental protocol for fitting validation.** *Left*: table summarizing main features of the experimental protocol. *Right*: spike raster depicting a typical 8s training input.

approached perfection on the same timescale. For the two-layer models, prediction and estimation needed about 4-8s of training data to reach perfection, and for the three-layer models, only the model with 20 synapses (the lowest number we tested) approached perfection by 8s. For the models with more synapses (25-40), prediction and estimation improved with increasing amounts of data but did not reach perfection within 8s. Though we have not yet tested performance on larger training data set sizes, it looks as though the three-layer models would also reach perfect performance within 16-32s, depending on the number of synapses. It is worth noting that in all cases the model achieves reasonable predictive performance before it attains reasonable parameter estimation performance, indicating that there is some degree of nonidentifiability in our model.

These results are encouraging for two reasons. First, they show that, despite the nonconvexity of our optimization problem and that we used an optimization procedure based on local derivatives, our estimation procedure performs successfully. Second, the amount of data needed for our models seems to be on the order of seconds, well within reach of current experiments. However, it is worth noting that the amount of data needed from real cells will probably be at least slightly larger than the amount needed from the synthetic models because out-of-class inference may have more (and nastier) local optima than the within-class inference we perform here.

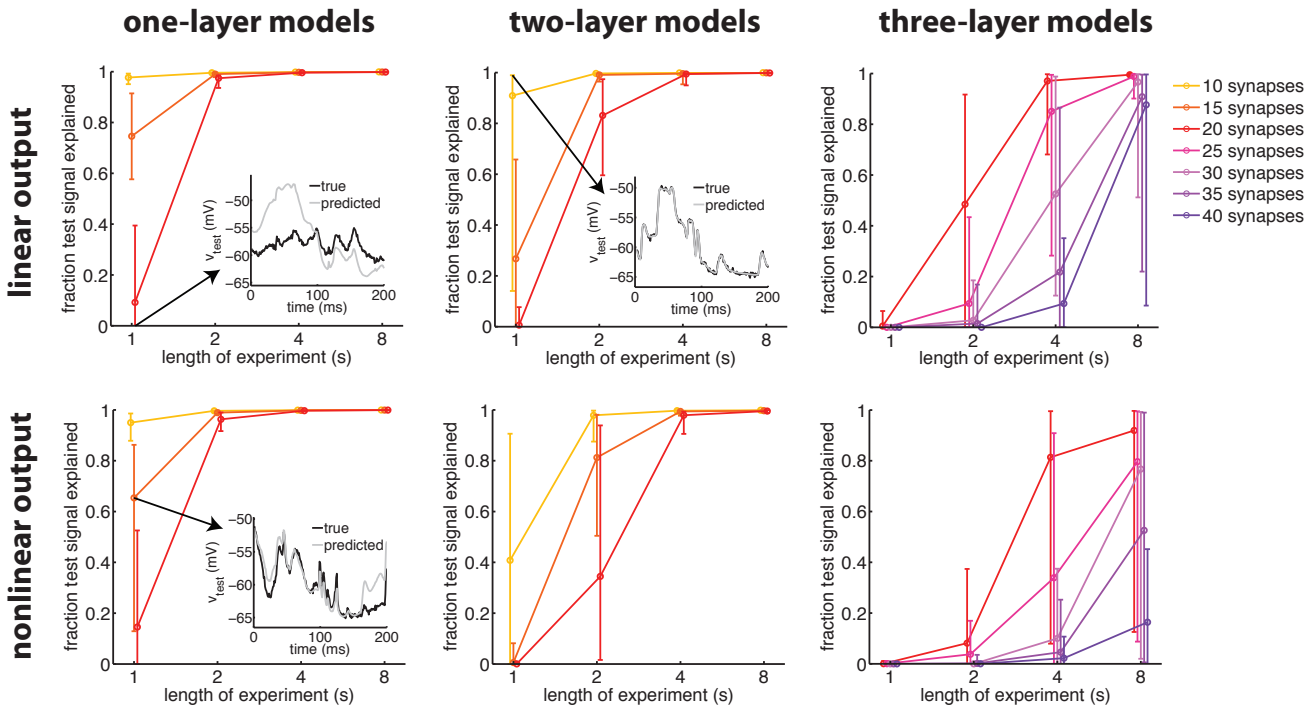

Figure 3.2: **Fitting validation.** Predictive performance was measured by the fraction of the test signal explained, defined as $f = 1 - \frac{\epsilon - \sigma}{s}$ where ϵ is the root-mean-square (rms) test set prediction error, s is the standard deviation (s.d.) of the test somatic membrane potential, and σ is the noise standard deviation. The horizontal axes show the amount of data in the training set (on a log scale) in seconds. Columns correspond to one-, two-, and three-layer models (from left to right) and rows correspond to models with linear (top) or nonlinear (bottom) root subunit. Lines of different colors correspond to different numbers of synapses (legend at top right). Circles indicate across-trial means of f. Error bars are 95% confidence intervals across trials. Insets show example membrane potential predictions to provide intuition for intepreting our test metric. The predictions are from a random 200ms interval of a trial with test performance indicated by the root of the black arrow and illustrate poor, reasonable, and excellent performance (counterclockwise from top left).

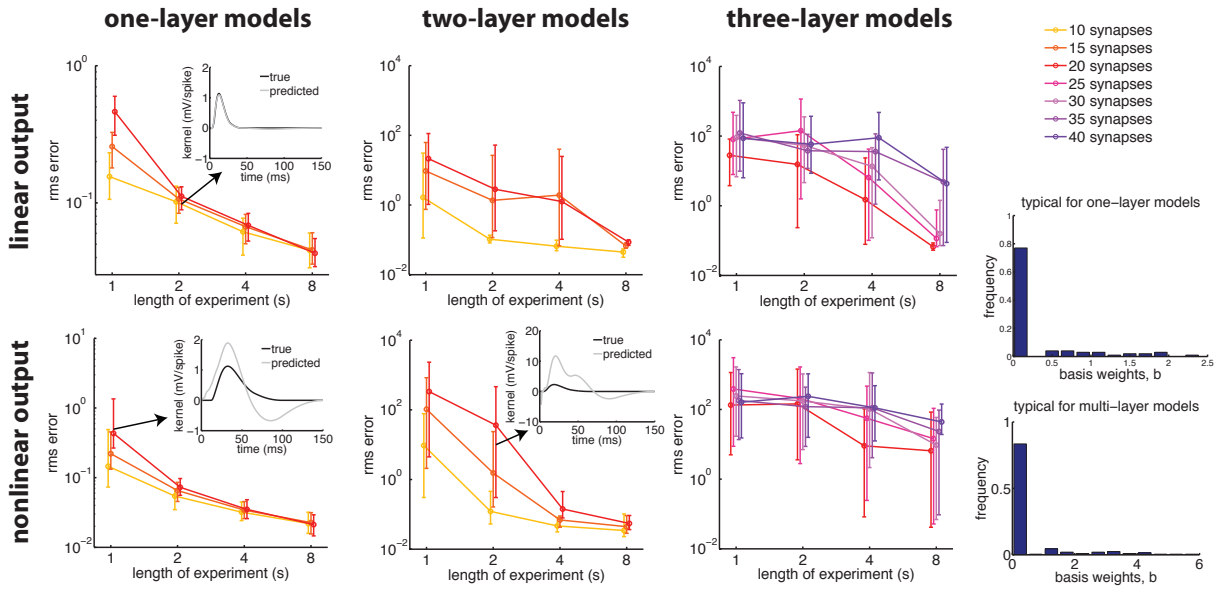

Figure 3.3: **Estimation validation.** Estimation performance is shown only for the synaptic kernel basis weights **b**, though estimation performance was similar or better for other model parameters. Estimation performance was measured by the rms error between the true and inferred basis weights. The horizontal axes show the amount of data in the training set (on a log scale) in seconds. Columns correspond to one-, two-, and three-layer models (from left to right) and rows correspond to models with linear (top) or nonlinear (bottom) root subunit. Lines of different colors correspond to different numbers of synapses (legend at top right). Circles indicate across-trial means. Error bars are 95% confidence intervals across trials. Two additional features are included to aid interpretation of the magnitude of rms errors. First, insets show example inferred kernels from a trial with estimation performance indicated by the root of the black arrow. The predictions are meant to illustrate poor, reasonable, and excellent performance (clockwise from bottom right). Second, the histograms at right illustrate typical distributions for the elements of **b** for one- (top) and multi-layer (bottom) models.

Chapter 4

Validation of model selection

Given that one of our goals is to make claims about the linear and nonlinear properties of real cells, we wanted to prove to ourselves that we could distinguish varying degrees of nonlinear behavior in synthetic models. If unsuccessful, this would seriously undermine our ability to distinguish between different models for real cells. If successful however, this would at least give us some confidence that we could identify varying degrees of nonlinear behavior under extended input trains with realistic statistics.

To perform model selection, we use cross-validation on a test set, that is we choose the model with highest predictive performance on a hold-out test set, as measured by the fraction test signal explained (see caption of figure 3.2 for definition). If two models perform equally well, we choose the simpler model.

To test our model selection capabilities, we fit 1L, 1N, 2L, and 2N models to data generated from each model and checked to see whether our model selection procedure identified the correct underlying model. Since we had already fit each of the four models to data generated from that same model in chapter 3, we needed only to fit the three remaining models to each data set. We concentrated only on the 8s training data sets, since we were interested in model selection using adequate amounts of data. In each case, we fit all 25 trials for each of the 3 numbers of synapses simulated (10, 15, and 20). For the cases in which a two-layer model was fit to one-layer data, we randomly assigned 5 synapses each to 2, 3, or 4 internal subunits, depending on the total of synapses; including the output subunit, there then 3, 4, or 5 subunits total.

The results of our model selection experiment are shown in figure 4.1. Most importantly, in every case, across models and number of synapses, our model selection procedure is successful in identifying the correct underlying model.

There are a couple features of figure 4.1 worth mentioning. To begin with, there is one case in which two models do equally well – for data from the 1L model, both the 1L and 1N models perform essentially perfectly. This occurs because a sigmoidal subunit can easily imitate a linear relationship by using small input weights (i.e. weights on subunits and synapses providing input to that subunit) and a bias which ensures the sigmoidal activation only receives small perturbations about its most linear regime (i.e. around zero activation). Indeed when we investigated the sigmoidal activation for the 1N model fit to 1L data, the parameters $\mathbf{w}$, $\mathbf{b}$, and $\boldsymbol{\theta}$ were set such that the 1N output sigmoidal was perturbed only slightly about zero activation and thus imitated a 1L model.

Also, one might wonder why, in all cases besides the 1N model fit to 1L data, the models more complex than the underlying model do more poorly than the underlying model. Shouldn't a 2L model be able to simulate a 1N model? And a 2N model a 2L model? There are at least two reasons why this is not true in this case. First, we did not attempt to search the space of all possible synapse assignments and number of internal subunits, so it is possible that other model architectures than the ones we used might have allowed the more complex models to fare better. For example, assigning all synapses in a two-layer model to a single internal subunit whose output is the sole input to an output subunit

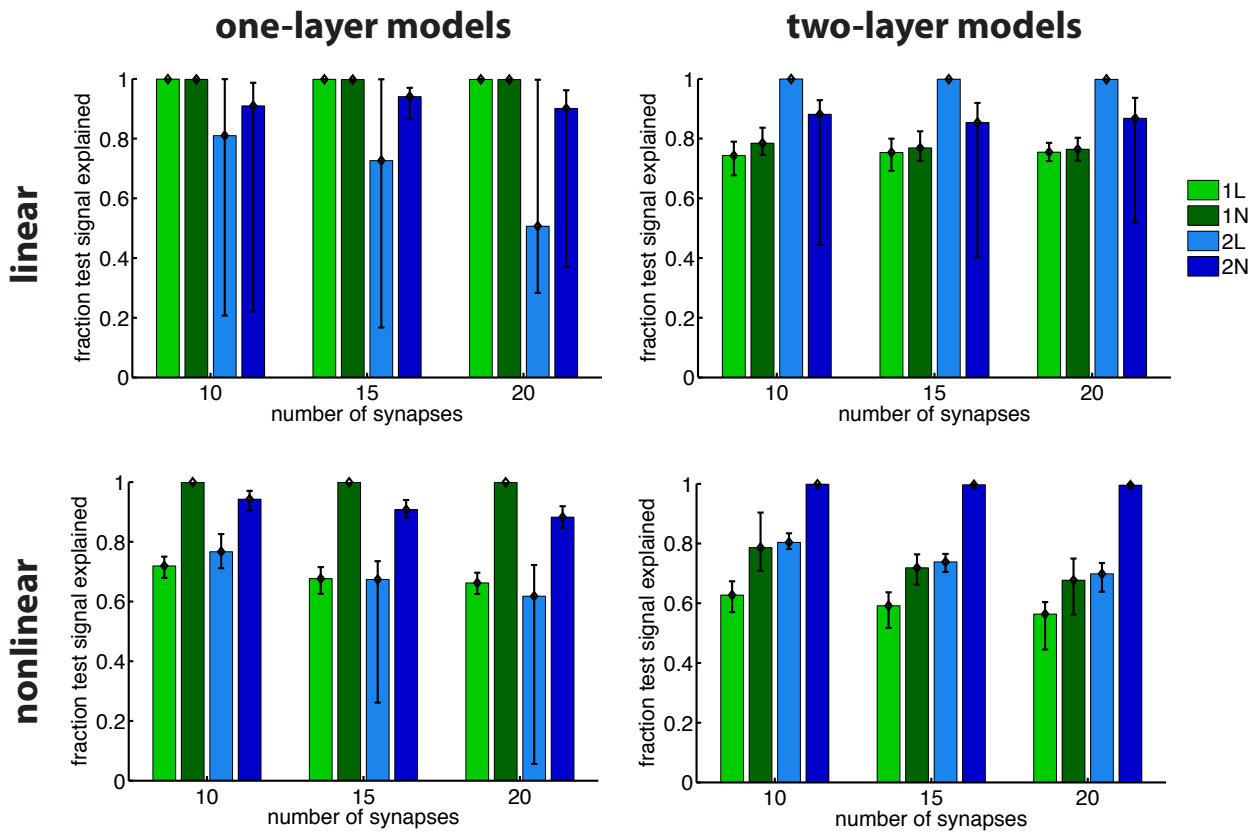

Figure 4.1: **Model selection validation.** Model selection is carried out using cross-validation on a test set. Bars indicate the fraction of the test signal explained, as defined in the caption of figure 3.2. Columns correspond to data generated from one- (left) and two-layer (right) models and rows correspond to data generated from models with linear (top) or nonlinear (bottom) root subunit. Bars of different colors indicate the model being fit to data (legend at top right). Bar heights indication across-trial means. Error bars are 95% confidence intervals across trials.

would make it much easier to imitate a one-layer model. Second, it is possible that the more complex (i.e. two-layer) models simply did not have enough training data. While 8s was certainly enough data in our within-class validation experiments (see chapter 3 and the appropriate bars in figure 4.1), it is possible that it was not enough data for out-of-class inference in the two-layer models. However, for the purposes of validating our model selection procedure, we do not need to worry about these issues. In all cases in figure 4.1, the underlying model performed essentially perfectly, so improvements in the performance of the more complicated models would not change the outcomes of our model selection.

Chapter 5

Fitting compartmental model simulation data

Assured that our fitting and model selection procedures were adequate, we next applied the hLN model to data from biophysically detailed multi-compartmental models. In particular, we set out to investigate (1) the computational subunits of cortical pyramidal neurons and (2) how the identification of these subunits depends on the stimulation protocol used. These experiments were motivated by the observation that many experiments investigating dendritic nonlinearities in real neurons use highly artificial stimuli without necessarily mentioning the effect this may have on the resulting nonlinearity properties identified. Indeed, the appropriateness of any input-output model of a nonlinear system will depend on the inputs used. For example, using small enough inputs to any (smooth) nonlinear system will make the input-output relationship appear linear.[1]

This investigation is further motivated by two recent studies. [Polsky et al 2004] noted that spatially and temporally clustered synaptic inputs lead to local nonlinearities in cortical pyramidal neurons, whereas inputs separated in either space (i.e. on different branches) or in time combine only linearly. However, as mentioned in chapter 1, that study used only paired inputs, rather than the extended spike train inputs that cells would receive *in vivo*, and heuristic measures of nonlinearity (i.e. peak EPSP). [Gasparini and Magee 2006] performed a similar study in hippocampal CA1 pyramidal neurons using more temporally extended inputs and reached similar conclusions. Nevertheless, their nonlinearity metrics were still heuristic (i.e. peak EPSP, firing rate, firing phase, occurrence of a dendritic spike), rather than based on model selection between input-output models predicting the full somatic membrane potential trace (as can be done with the hLN model).

Building on [Polsky et al 2004] and [Gasparini and Magee 2006], we endeavored to demonstrate the dependence of identified dendritic nonlinearities on the spatial and temporal statistics of their synaptic inputs using spike trains of more realistic timescales, as well as a more principled metric for judging nonlinear behavior (i.e. model selection between increasingly complex hLN models).

In all cases, we used the cortical pyramidal neuron model with AMPA and NMDA synapses described in [Branco et al 2010]. In brief, the model includes a soma, an axon, 69 dendritic compartments, and the following active channels: Hodgkin-Huxley type Na^+ and K^+ channels, M-type K^+ channel (slow, non-inactivating), Ca^{2+}-dependent K^+ channel, high-threshold Ca^{2+} current, and T-type Ca^{2+} channel. Simulations were all performed by Balázs Ujfalussy using the NEURON simulation environment [Hines and Carnevale 1997].

In the first four experiments (sections 5.1-5.3), the active currents were not included in the model and all nonlinear effects were mediated by either NMDA spikes (superlinear) or the reduction of the synaptic driving force (sublinear) at large, synchronous inputs.

[1] This is the observation behind a common technique in applied mathematics for analyzing nonlinear systems called "linearization."

5.1 Experiment 1: clustered vs. distributed synapses, 20Hz inputs

Our first experiment sought to demonstrate a simple point – that the identification of dendritic nonlinearities would depend on both (1) synapse location and (2) correlations in the stimulus (i.e. input spike trains). To do so, we stimulated 10 synapses that were either (a) clustered on a single basal dendrite or (b) distributed across 10 different branches. Our stimulus protocol was designed to allow parametric variation of correlations with fixed marginal statistics of $r = 20$Hz stimulation to each synapse. More specifically, the input spike train to each synapse consisted of two Poisson components: (1) an independent component with rate $(1-\alpha)\,r$, where $0 \leq \alpha \leq 1$, and (2) a shared component with rate αr. While the independent component was private to each synapse (i.e. 10 different trains were generated), the shared component was common to all synapses (i.e. only one was generated). Thus, α denotes the fraction of shared (synchronous) spikes, $\alpha = 0$ corresponds to independent inputs, and $\alpha = 1$ corresponds to synchronous inputs only. The spike train to each synapse then consisted of the sum of the independent and shared spike trains, thresholded to eliminate multiple spikes at a synapse in a single time bin.[2] A small (1ms) jitter was added to the shared spikes. Each condition (i.e. clustered vs. distributed, value of α) was repeated for 10 trials, each with 16s of training data and 4s of test data. The experimental protocol is summarized in figure 5.1.

The results of this experiment are shown in figure 5.2. For distributed synapses, the linear model performs very well regardless of input synchrony (left plot) and the nonlinear model offers little to no improvement (left and right plots). For clustered synapses, the situation is quite different. When the inputs are independent ($\alpha = 0$), the nonlinear model offers little improvement over the linear model (left and right). As input synchrony increases however, the linear model becomes progressively worse (left), while the nonlinear model maintains high performance (left), so that the nonlinear models performs progressively better than the linear model (left and right). Across all levels of synchrony, prediction seems to be more difficult for the clustered case than for the distributed case, as both models perform significantly worse.

These results indicate that both the choice of synapse locations and stimulus statistics can play a major role in the identification of dendritic nonlinearities. When synapses are chosen from throughout the dendritic tree, synaptic integration may appear linear, regardless of the stimulation protocol. When synapses are chosen from a tightly clustered group on a single branch, however, synaptic integration may appear linear for independent stimulation but nonlinear for correlated (synchronous) stimulation. Importantly, these results also suggest that the hLN model can capture quite well the behavior of a detailed biophysical compartmental model receiving extended inputs with varying statistics. Indeed, across all levels of synchrony, the best hLN model predicts about 90 or 95% of the test signal for clustered and distributed synapses, respectively.

However, upon closer examination of our results, we found an important caveat – the relatively strong 20Hz stimulation which we applied in this experiment left all of the NMDA synapses in the desensitized state (i.e. unresponsive to inputs) for virtually the entire duration of the experiment. Thus, the results of this experiment speak only to a passive dendritic tree. The reason we still found nonlinear behavior under these conditions is due to the reduction of the synaptic driving force during the large, local depolarizations caused by the synchronous events.

[2] While this thresholding step does cause the marginal statistics and fraction of shared spikes to vary from the numbers we intended, the deviations are neglible except for firing rates much larger than the ones we use. Specifically, the thresholding reduces both r and α by $\alpha\,(1-\alpha)\,r^2$, where r is in spikes per ms, which even in the worst case scenario (i.e. $\alpha = 0.5$) corresponds to errors of 1 part in 200 and 1 part in 5000 for r and α, respectively.

number of synapses	10
location of synapses	distributed or clustered
input firing rate	20Hz
correlations?	yes, parameterized by α
conductances	passive only (including NMDA)
size of training set	16s
size of test set	4s
trials per condition	10

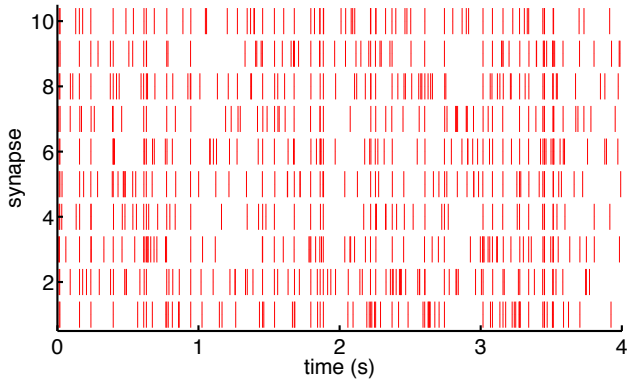

Figure 5.1: **Clustered vs. distributed synapses, 20Hz inputs - summary of experimental protocol.** *Left*: table summarizing main features of the experimental protocol. *Right*: spike raster depicting 4s of a typical training input with $\alpha = 0.4$. Although we show results in figure 5.2 only for values of α up to 0.2, we use a higher value here for easier visualization of the shared spike events.

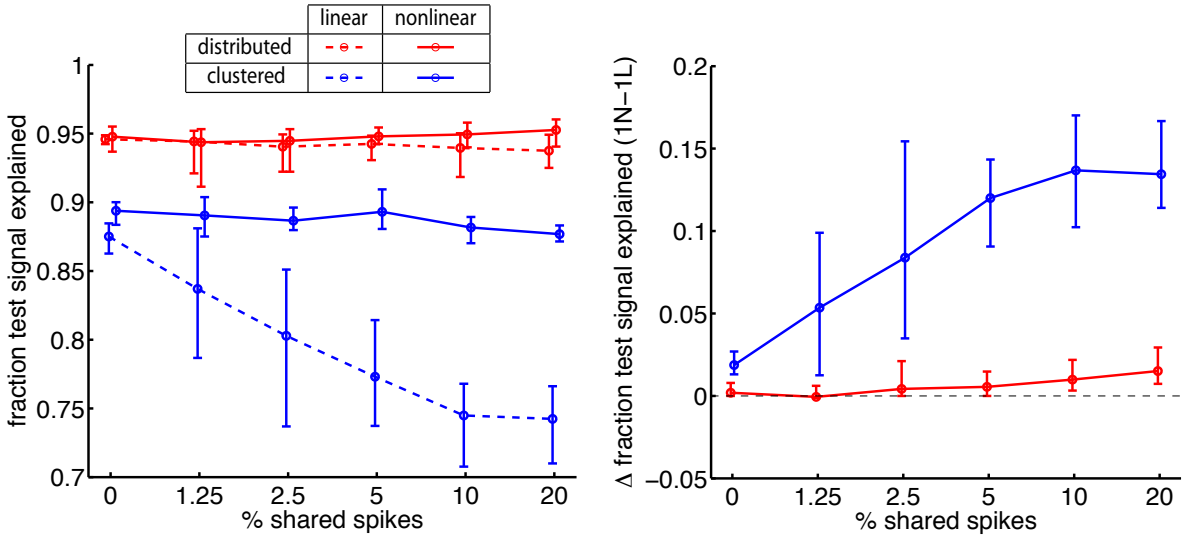

Figure 5.2: **Clustered vs. distributed synapses, 20Hz inputs - fitting results.** Model performance was measured by the fraction of the test signal explained, defined as $f^* = 1 - \frac{\epsilon}{s}$ where ϵ is the root-mean-square (rms) test set prediction error and s is the s.d. of the test signal (i.e. the test set somatic membrane potential). This definition differs from that of f in the caption to figure 3.2 because in this case we do not know the true level of noise. The horizontal axes show the fraction of shared spikes α in percentage (on a log scale). The left plot shows f^* for the 1L (linear) and 1N (nonlinear) models fit to data from the stimulation of clustered or distributed synapses. The right plot shows the trial-by-trial difference between f^* for the 1L and 1N models. A dotted black line at zero is included for reference. Points above the black line indicate better performance of the 1N model over the 1L model. Circles indicate across-trial means. Error bars are 95% confidence intervals across trials.

5.2 Experiment 2: clustered vs. distributed synapses, 2Hz inputs

In order to obtain results for an active dendritic tree (i.e. with active NMDA synapses), we reran the experiment described in section 5.1 with $r = 2$Hz. Since there were fewer spikes, we also increased the amount of training and test data to 32s and 8s, respectively. The experimental protocol (and differences from that of the experiment in section 5.1) are summarized in figure 5.3.

The results are shown in figure 5.4. For distributed synapses and for clustered synapses with independent inputs, the results were essentially identical to the experiment with higher input firing rate in the last section. For clustered synapses with synchronous inputs, it was still the case that the 1N model offered a significant advantage over the 1L model, suggesting a nonlinear integration mode for spatially and temporally correlated inputs. However, in addition, both the 1L and 1N models performed significantly worse for the lower firing rate. While we expected the 1L model to perform worse since active NMDA synapses should make the cell more nonlinear, we were surprised to see the 1N model also perform significantly worse.

A closer investigation of the prediction errors reveals a few possible sources for how and why the 1N model fails (figure 5.5). First, the fitted synaptic kernels typically display a lower peak and slower decay than the true kernels (figure 5.5, left). Second, during the large depolarizations caused by a wave of synchronous inputs, the fitted membrane potential typically peaks more quickly and fits the decay of the true membrane potential only by including an extra "bump" during the decay (figure 5.5, left). Combined, these two observations indicate that the model is sacrificing accuracy in fitting the time course of the synaptic kernels in order to achieve greater accuracy in fitting the time course of the large depolarization events. This can be understood by noting that our use of maximum likelihood fitting and a Gaussian noise model corresponds to using a least-squared error (LSE) criterion, which penalizes large errors much more strongly than small errors. It is then not surprising that the model contorts itself to achieve accuracy on the larger events (where correspondingly large errors are possible) at the cost of decreased accuracy for the small events (where errors are typically much smaller).

This investigation also reveals four potential directions for improving the hLN model. First and foremost, including a mechanism for extending the time course of the response during large depolarizations might help the model fit the data without sacrificing accuracy on the synaptic kernels. We discuss two such mechanisms – an output feedback filter and latent variable models - in sections 6.2.4 and 6.2.5, respectively. Second, including a sparsity-inducing penalty for the synaptic kernel basis weights might discourage the unrealistic extra "bumps" used to fit the large depolarization events. We discuss this option further in section 6.1.3. Third, including more single-synapse stimulation events in the training set or, equivalently, placing a larger weight on the training error for single-synapse stimulation events during fitting would place a higher emphasis on fitting the synaptic kernels more accurately. Fourth, altering the basis set for the synaptic kernels might allow more accurate fits. We discuss this option further in section 6.2.1.

5.3 Experiment 3: two branches, within- and across-branch synchronization

Given suggestions by others that cortical pyramidal neurons function as a "two-layer network" [Poirazi et al 2003a, Poirazi et al 2003b, Polsky et al 2004], we next sought to identify two-layer behavior using the hLN model. To do so, we stimulated 20 synapses, 10 each on 2 different branches. The input spike trains again were the sum of two Poisson components yielding a 2Hz firing rate for each synapse: (1) an independent component and (2) a shared component. Each shared spike event was this time shared across a random subset of the synapses (8 of 20), regardless of location. Thus, the shared events included ones in which one branch was stimulated much more strongly than the other (asymmetric), as well as events in which both branches were stimulated evenly (symmetric). We chose this stimulus

number of synapses	10
location of synapses	distributed or clustered
input firing rate	*2Hz*
correlations?	yes, parameterized by α
conductances	passive only (including NMDA)
size of training set	*32s*
size of test set	*8s*
trials per condition	10

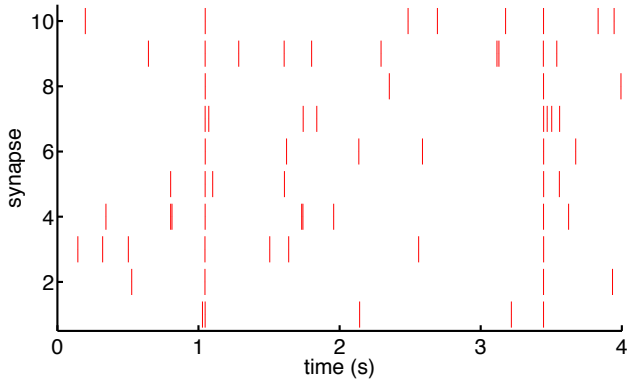

Figure 5.3: **Clustered vs. distributed synapses, 2Hz inputs - summary of stimulation protocol.** *Left*: table summarizing main features of the experimental protocol. Features that differ from the experiment summarized in figure 5.1 are shown in italics. *Right*: spike raster depicting 4s of a typical training input with $\alpha = 0.4$.

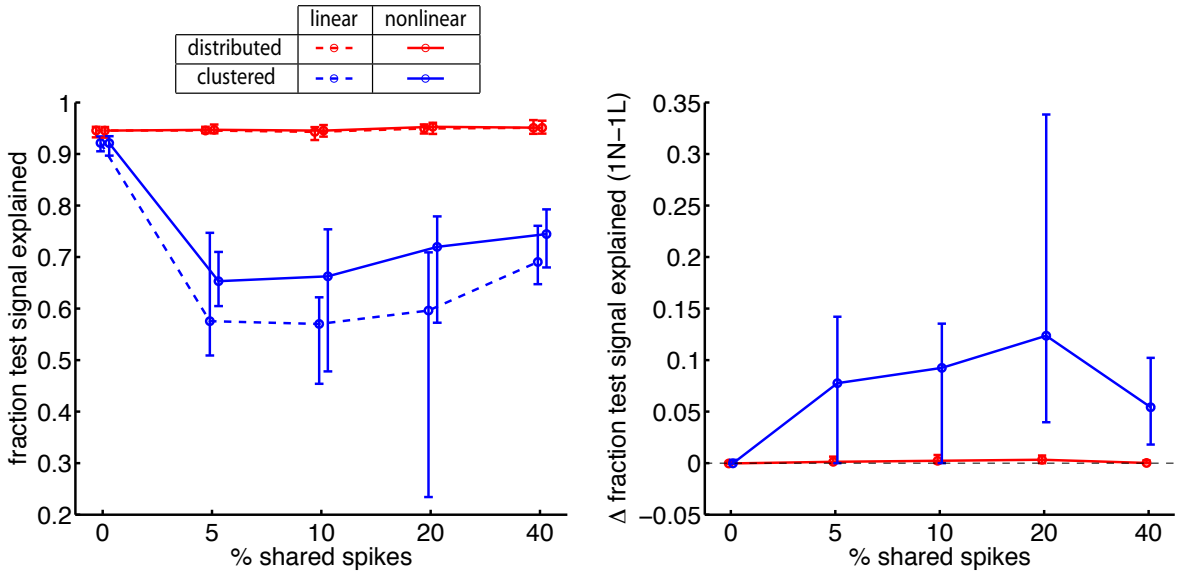

Figure 5.4: **Clustered vs. distributed synapses, 2Hz inputs.** See caption for figure 5.2.

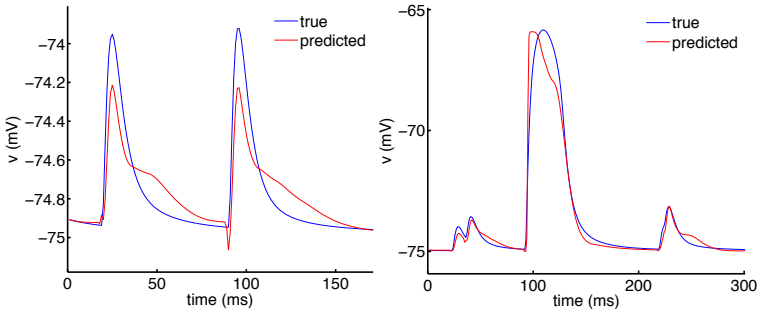

Figure 5.5: **Typical prediction errors.** Two short sections of the training error from the same trial of a 1N model fit to simulation data of the type described in section 5.2 with $\alpha = 5\%$ are shown to emphasize two typical prediction errors for the hLN model. *Left*: typical errors for the synaptic kernels. *Right*: typical errors for the large depolarization events triggered by synchronous inputs. Note the very different y-axis scales for the left and right plots.

protocol to elicit two-layer behavior based on the intuition that the asymmetric events would be above threshold for the local nonlinearities, while the symmetric events and independent Poisson inputs would be below threshold and elicit only linear integration, leading the 2L model to significantly outperform the 1L and 1N models. We again systematically varied the fraction of shared events α from fully independent ($\alpha = 0$) to nearly only shared events ($\alpha = 0.8$), as described in section 5.1, and again added a small (1ms) jitter to the shared spikes. For each α, we ran 12 trials, each with 32s of training data and 8s of test data. The experimental protocol is summarized in figure 5.6.

The results are shown in figure 5.7. Contrary to our expectations and previous work by others [Poirazi et al 2003a, Poirazi et al 2003b, Polsky et al 2004], we found the behavior of the cortical pyramidal neuron under this stimulation protocol to be decidely one-layer - while the 1N model offered a small performance improvement over the 1L model, the 2L and 2N model offered little to no additional improvement.

There are several reasons why this set of experiments might have suggested 1N behavior, while the experiments in section 5.1 suggested two-layer behavior. I believe the most likely candidate however is that there may have been two few asymmetric shared events in the inputs. In such a situation, even if we replaced the cell with a synthetic 2N hLN model, a 1N model could do very well at fitting the data. To see this, note that a 1N model could be fit to the symmetric shared events, since these could be linearly integrated by the first layer of the 2N model and then passed through the final nonlinearity to yield 1N behavior. Similarly, the asychronous independent Poisson inputs might be so weak as to elicit linear integration in both layers, which could also be captured by a 1N model.

5.4 Experiment 4: two branches, with active conductances

In order to ensure that there was a more uniform distribution of symmetric and asymmetric shared events and that the integration regime of the simulated cell was more fully explored, we used a more contrived stimulation protocol designed specifically for this purpose. Our protocol involved stimulating 12 synapses each on 2 different branches (24 synapses total) with several types of stimulation, with each synapse participating in each type of stimulation at least once. The stimulation types were: (a) each synapse activated alone (24 events), (b) all 12 synapses on one branch activated together (2 events), (c) 6 randomly selected synapses on one branch activated together (100 events, 50 for each branch), (d) 3 randomly selected synapses on each branch activated together (100 events), and (e) 6 randomly selected synapses on each branch activated together (100 events). We chose these stimulation types to explore the full integrative regime possible for 2 branches. In other words, a 1L, 1N, 2L, and 2N model would each show very different behavior under this range of inputs. These stimulus event types

number of synapses	20
location of synapses	10 each on 2 branches
input firing rate	2Hz
correlations?	yes, within- and across-branch, parameterized by α
conductances	passive only (including NMDA)
size of training set	32s
size of test set	8s
trials per condition	12

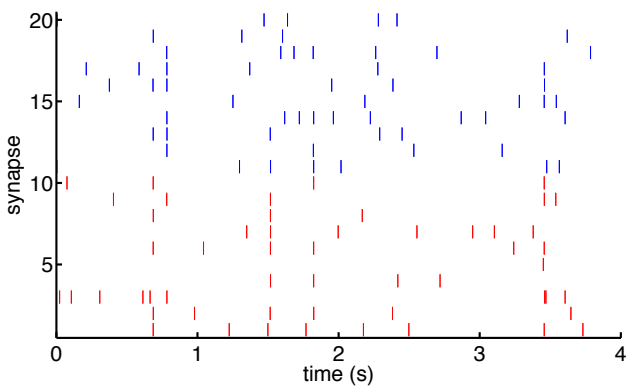

Figure 5.6: **Two branches, within- and across-branch synchronization - summary of experimental protocol.** *Left*: table summarizing main features of the experimental protocol. *Right*: spike raster depicting 4s of a typical training input with $\alpha = 0.4$. Spike color indicates dendritic branch membership – synapses 1-10 (red) belong to one branch while synapses 11-20 (blue) belong to another.

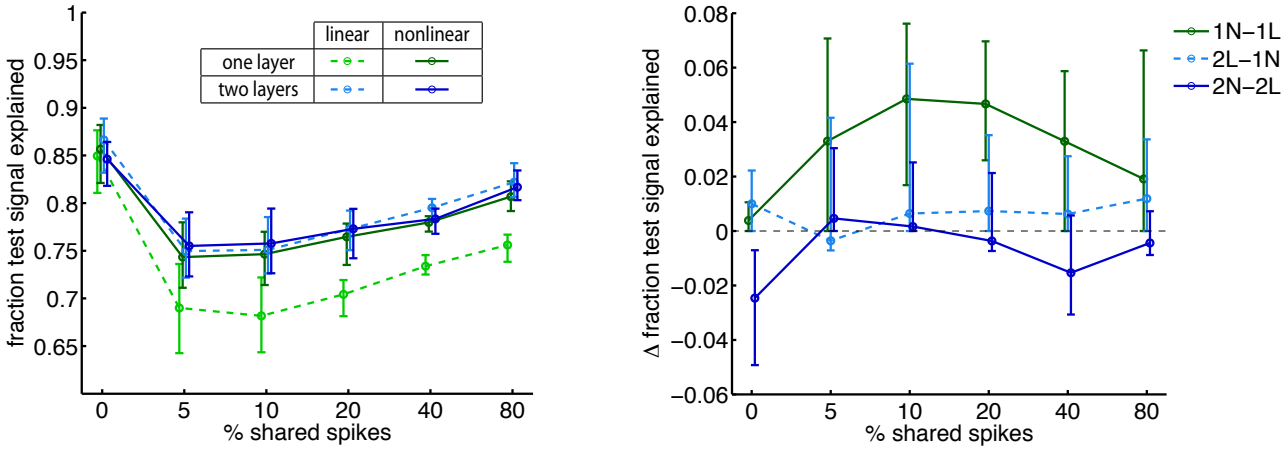

Figure 5.7: **Two branches, within- and across-branch synchronization.** Model performance was measured by the fraction of the test signal explained f^*, as defined in the caption for figure 5.2. The horizontal axes show the percentage of shared spikes α (on a log scale). The left plot shows f^* for all models. The right plot shows the trial-by-trial difference between f^* for three pairs of models: 1N and 1L, 2L and 1N, and 2N and 2L. These numbers are meant to suggest the improvement offered by each additional level of complexity in the hLN model. A dotted black line at zero is included for reference. Points above the black line indicate better performance of the more complex model. Circles indicate across-trial means. Error bars are 95% confidence intervals across trials.

as well as compartmental model and fitted hLN model responses are depicted in figure 5.8 (top right and bottom).

We also reinstantiated the active conductances described in the beginning of chapter 5. While the authors of the cortical pyramidal neuron that we used claimed that their experimental results could be captured with the NMDA and AMPA conductances alone [Branco et al 2010], we wanted to see the effect of including active conductances. By adding the active conductances however, the compartmental model was now capable of generating spikes. Since our model was intended for only subthreshold membrane potential, we sought to remove the occurrence of somatic action potentials by reducing the maximal conductance of the Hodgkin-Huxley sodium and potassium channels in the soma and axon to their dendritic values ($g_{Na} = 40\frac{\text{S}}{\text{cm}^2}$, $g_K = 30\frac{\text{S}}{\text{cm}^2}$).

With these modifications, we ran 2 trials with 50s of training and test data each (the training data for each trial was used as the test data for the other). The experimental protocol is summarized in figure 5.8.

The results are shown in figure 5.9. Of all experiments discussed in this report, the 1L model performed worst for this experiment, indicating highly nonlinear behavior. Much of this nonlinear behavior was captured by the 1N model (i.e. an increase in f^* of >30% over the 1L model). However, there was still a significant increase in test performance for the 2N model over the 1N model (~7%). As in the other experiments (see e.g. figure 5.5), we see again that all models contort themselves to fit the events of largest amplitude at the cost of predicting more poorly smaller amplitude events (bottom of figure 5.9).

Perhaps counter to intuition, the 2L model performed significantly worse than the simpler 1N model, nearly as poorly as the 1L model. The reason for this is that, as in the other experiments in which the two-layer model was used, we fit the 2L model with only one architecture - the one corresponding to the known branch membership of the synapses. With other architectures, it may of course have been possible to achieve better performance. For example, a 2L model with all synapses belonging to a single internal subunit corresponds exactly to the 1N model and would thus have achieved the same level of performance. With the architecture we used however, the 2L model can capture local nonlinearities (e.g. in the dendrites) but cannot capture global nonlinearities (e.g. in the soma). Similarly, the 1N model can capture global but not local nonlinearities and the 2N model can capture both types. Thus, we can also read the results in figure 5.9 as indicating the presence of a strong global nonlinearity (captured by the 1N and 2N models) with weaker local nonlinearities (captured by the 2L and 2N models).

In summary, although our stimulation protocol was designed explicitly to encourage two-layer behavior, the performance gains for the two-layer models over the one-layer models were surprisingly meager. While the performance advantages may be larger for other cell types, other compartmental models of the same cell, other stimulation protocols, or improved versions of the hLN model (see chapter 6), our results still suggest that two-layer behavior in cortical pyramidal neurons is quite difficult to elicit using extended spike train inputs. This is in contrast to previous results using briefer inputs and more heuristic measures of nonlinearity [Poirazi et al 2003a, Poirazi et al 2003b, Polsky et al 2004].

number of synapses	24
location of synapses	12 each on 2 branches
input firing rate	varies widely in time
correlations?	yes, within- and across-branch, depending on event type
conductances	active and passive
size of training set	50s
size of test set	50s
trials per condition	2

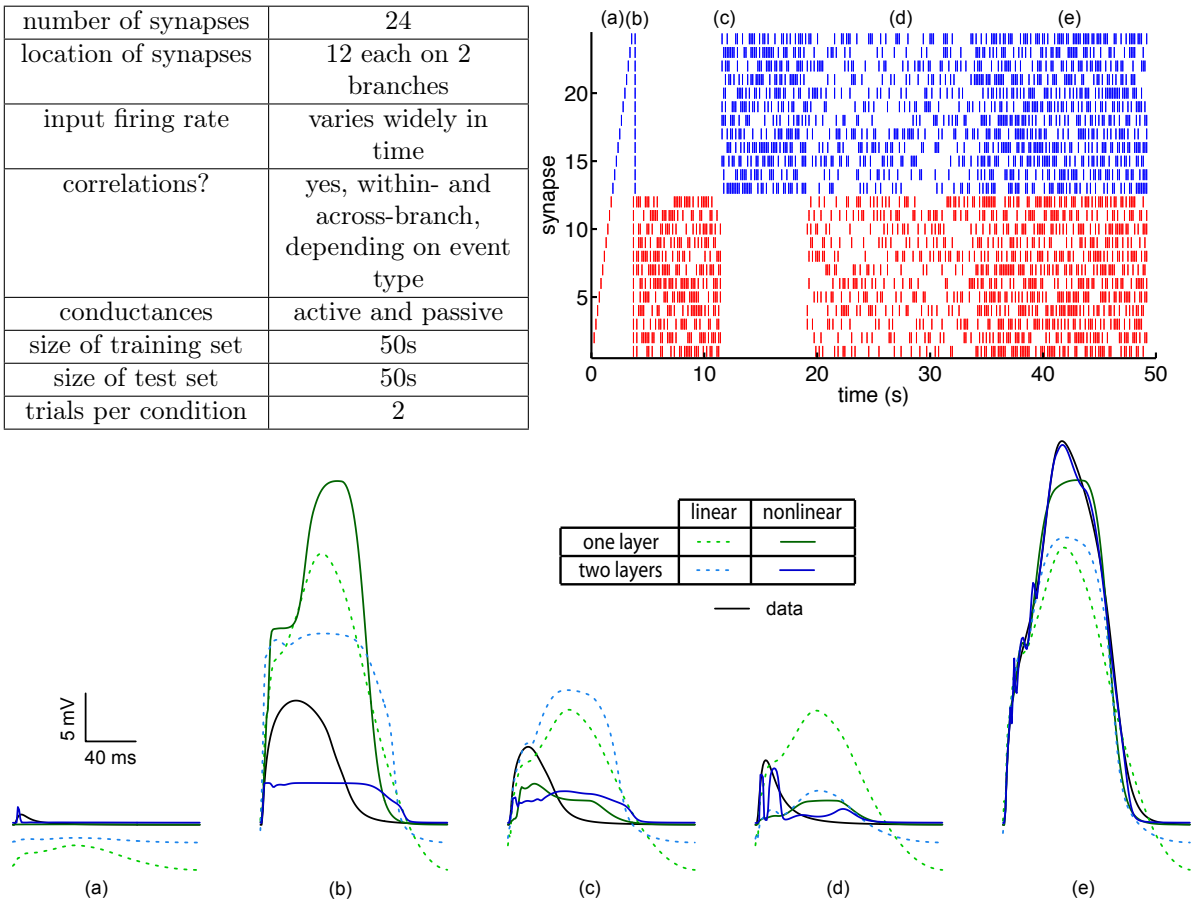

Figure 5.8: **Two branches, active conductances - summary of experimental protocol.** *Top left*: table summarizing main features of the experimental protocol. *Top right*: spike raster depicting a typical training input. Spike color indicates dendritic branch membership – synapses 1-12 (red) belong to one branch while synapses 13-24 (blue) belong to another. Letters above the plot refer to the stimulus event types described in the main text. *Bottom*: Average fitted responses to the five different stimulus event types (described in the main text and illustrated at top right) for the four different models as well as for the compartmental model ("data").

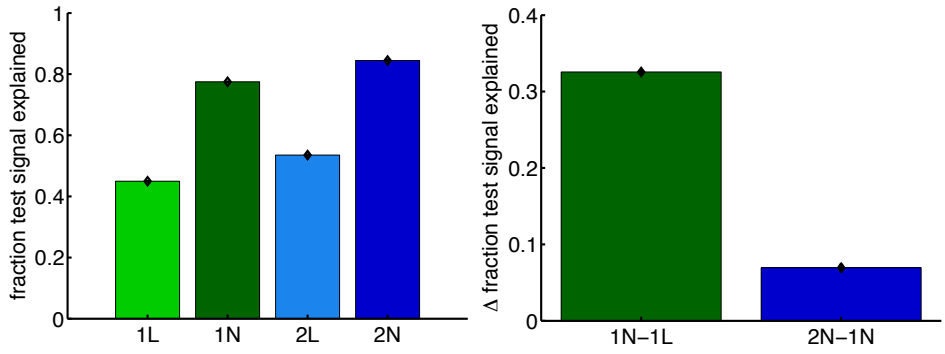

Figure 5.9: **Two branches, with active conductances.** Model performance was measured by the fraction of the test signal explained f^*, as defined in the caption for figure 5.2. The left plot shows f^* for all models. The right plot shows the trial-by-trial difference between f^* for two pairs of models: 1N and 1L, and 2N and 1N. These numbers are meant to suggest the improvement offered by each additional level of complexity in the hLN model. Bar heights indicate across-trial means. Error bars are 95% confidence intervals across trials (but are too small to be seen).

Chapter 6

Discussion

6.1 Improving parameter estimation

The estimation procedure we use here leaves plenty of room for improvement. We discuss several such directions in this section.

6.1.1 Structure learning

The crude form of model selection which we use here is problematic in that it requires us to explicitly specify the model architectures (i.e. synapse-to-subunit and subunit-to-subunit connectivities) that we wish to test. A more principled approach would be to automatically learn the appropriate architecture from data, a process usually referred to as "structure learning" in the machine learning literature. Given that the space of such architectures is enormous, approximate inference procedures for efficiently searching it would be necessary. While we have not yet attempted to solve this problem, inspiration for approaches might be found in the literatures on hierarchical clustering and diffusion trees [Duda et al 2001, Heller and Ghahramani 2005, **?**] or Bayesian nonparametrics (e.g. Chinese restaurant and Indian buffet processes [Griffiths and Ghahramani 2011]).

6.1.2 Active learning

For all experiments described here, the data was produced in its entirety before any model fitting was begun. A more advanced approach would be to change the stimulation protocol online and automatically in order to optimize the informativeness of the experiment. This approach is known as "optimal experimental design" [Huggins and Paninski 2012] or "active learning" [Houlsby et al 2011]. While it is not of much practical value when the source of data is a computational model and data is easy to obtain, active learning can be especially helpful when the origin of the data is a real cell and data is thus more difficult to obtain. In fact, for glutamate uncaging, the experimental approach most appropriate for testing the hLN model in real cells, minimizing the length of the experiment can be crucial, as cells are easily damaged and killed during extended stimulation.

6.1.3 Sparsity-inducing regularization of synaptic kernel basis weights

As noted at the end of section 5.2, fitting the current version of the hLN model to data from compartmental models typically results in synaptic kernels with an extra "bump" resulting from the combination of two or more basis functions with disparate time courses. While these extra bumps seem to help the model fit the temporally extended responses seen during large depolarizations, they are typically poor models for the synaptic kernels. Along with changes to the model class to accommodate the

temporally extended responses, discouraging synaptic kernels with multiple bumps could lead to more realistic model fits.

One approach to doing so is to modify the objective function in our fitting to include both the log likelihood (eqn 2.7) and a sparsity-inducing penalty on the synaptic kernel basis weights [Bach et al 2012]. Examples of such penalties include the L1 ($\lambda \sum_{i=1}^{N_{\text{syn}}} |\mathbf{b}_i|$) and L2 norms ($\lambda \sum_{i=1}^{N_{\text{syn}}} \|\mathbf{b}_i\|$), where λ is a number describing the strength of the regularization and is typically chosen by cross-validation.

6.1.4 Alternative objective functions

As mentioned in section 2.3, using ML fitting with an additive white Gaussian noise model is equivalent to choosing the parameters which minimize the squared training error. One might argue, however, that a more appropriate objective function would penalize more strongly prediction errors close to the firing threshold of the cell or even define errors in the space of firing rates or spike times.[1]

6.2 Expanding the model class

In addition to improving our estimation procedure, there are several directions for expanding our model class which might address some of the shortcomings observed in the experiments above (see e.g. the end of section 5.2).

6.2.1 Alternative basis functions

The set of basis functions we used here (see eqn 2.4 and [Pillow et al 2008]) were chosen for the reasons described in section 2.2. However, they were originally introduced for modeling stimulus sensitivities in sensory neurons [Pillow et al 2008]; it is more common to model synaptic kernels using alpha functions of the form $f_m(t') = \frac{t}{\tau_m} e^{-\frac{t}{\tau_m}}$ where τ_m controls the timescale of the response. It is possible that an alpha function basis would be more natural and lead the hLN model to perform better in fitting data from compartmental models as well as real cells.

6.2.2 Alternative noise models

The additive white Gaussian output noise which we use here (see eqn 2.3) is perhaps the simplest noise model one might imagine, but it is clearly not the most realistic.

For example, some researchers observe that dendritic spikes are generated stochastically [Diba et al 2006].[2] A more appropriate noise model for the hLN might include a stochastic dendritic spike initiation step as well. This could be accommodated by, for example, replacing the internal LN subunits with LNP subunits and including another set of filters in the next layer to model the dendritic spikes.

As another example, it is widely believed that synaptic transmission is stochastic as well. Thus, when fitting data from real cells in which the inputs are spiking events detected in presynaptic cells, it may be necessary to include a stochastic step allowing for synaptic transmission failure.[3]

6.2.3 Additional nonlinearities

In all of the work discussed here, we used for the subunit output nonlinearity $g(\cdot)$ (see eqn 2.1) a sigmoid. Ideally however, we would infer the correct form of the nonlinearity from the data. This could be accomplished either by (1) selecting the nonlinearity from a small set of candidate nonlinearities

[1] This would of course require the hLN model to be extended to model firing rates or spikes; see chapter 7 for a brief discussion of this possibility.

[2] Note that the compartmental model we used was deterministic (as most are), so in the experiments discussed here a stochastic dendritic spike mechanism was not necessary.

[3] For compartmental models and glutamate uncaging experiments (in which postsynaptic sites are stimulated directly), this again does not seem necessary.

(e.g. linear-step-linear, soft linear, or linear + quadratic + sigmoid [Poirazi et al 2003a]) or (2) inferring the nonlinearity using a more flexible approach, such as splines, piecewise linear functions, or Gaussian processes.

6.2.4 Output feedback filter

As mentioned at the end of section 5.2, the current version of the hLN model has trouble capturing the extended time course of the membrane potential response during large depolarizations (see e.g. figure 5.5) without sacrificing accuracy on the synaptic kernels. One way of accommodating this phenomenon directly would be to introduce a feedback term to the model which passes the observed membrane potential through a thresholding nonlinearity followed by a temporally extended filter and adds the result to the membrane potential prediction. This idea is similar in spirit to the post-spike filter which is often included in GLMs in order to capture behaviors such as refractoriness and bursting [Pillow 2007, Pillow et al 2008].

6.2.5 Latent variable models

Another way to capture the extended response time course mentioned above is to augment the hLN model with a set of latent variables. Latent variables are used to model the dynamics of unobserved quantities and have been used in neuroscience to, for example, model unrecorded shared input influencing population responses [Macke et al 2011]. In our case, they might be used to model the slow dynamics of channels and ions. Latent variables could also be used to model other neural phenomena which become especially important in experiments with real cells, such as synaptic facilitation and depression, a drifting baseline membrane potential, or changes in recordings due to electrode movement.

6.3 Additional experiments

There are at least two main important directions for future experimental applications of the hLN model.

6.3.1 Natural synaptic inputs

The first is to examine the integrative properties of dendrites under more natural stimulation. Unfortunately, at the time of this writing very little is known about *in vivo* synaptic input statistics [Jia et al 2010, Chen et al 2011, Varga et al 2011]. Nevertheless, we have also been working on generating plausibly more realistic synaptic inputs using a model described in [Ujfalussy and Lengyel 2011]. The basic idea is to (1) simulate spatially and temporally correlated presynaptic membrane potentials using a multivariate Ornstein-Uhlenbeck (mOU) process and (2) produce spikes from those membrane potentials by passing them through a nonlinearity (typically exponential) and using the result as the rate for a Poisson spike-generating process. The resulting stochastic process is called the "mOU-NP" model and is capable of simulating many experimentally observed phenomena. For example, with the inclusion of a switching process, the "switching-mOU-NP" model can simulate up and down states.

6.3.2 Experiments with real neurons

Perhaps most importantly, it is necessary to both test the appropriateness of the hLN model and examine the integrative properties of dendrites using data from real neurons. At the moment, the most relevant experiments to our knowledge involve glutamate uncaging *in vitro*. To this end, we are currently collaborating with Tiago Branco, Michael Häusser, and Judit Makara, though no experimental data was ready for presentation at the time of writing.

Chapter 7

Conclusions

We have introduced the hLN model for dendritic computation as a simple, analytically tractable, and interpretable model which can be efficiently fit directly to existing experimental data. Moreover, it is capable of handling dynamic inputs, modeling arbitrarily complex dendritic trees, and being judged by its predictiveness of full somatic membrane potential traces - all properties which previous models have lacked. We have introduced a parameter estimation procedure for fitting our model to experimental data and validated the success of that procedure on synthetic data sets. We have also introduced criteria for model selection and demonstrated its success on synthetic data sets. Finally, we have demonstrated that the hLN model can be fit to data from biophysically detailed compartmental models of single neurons and used the model to demonstrate that conclusions about local dendritic computation depend strongly on the stimulation protocol.

Yet, it is clear that there is room for improvement in the hLN model. We have introduced several directions for such improvement, among the most pressing of which are a more principled approach to model selection, more appropriate synaptic kernel basis functions, an output feedback filter, and the addition of latent variables. Other directions may be suggested once the model is tested on data from real cells. Also, while we have chosen to focus on predicting somatic membrane potentials, it seems relatively straightforward to use the model to predict spikes, either by fitting the output of the hLN model to firing rates or by adding a spike-generating mechanism to the model output, as is done in the GLM and LNP models.

Finally, since we began with questions regarding "the fundamental computational unit" of the brain, we should return to them. What is clear from the work described here is that the answers to these questions depend on the input statistics each cell receives – until those statistics are better characterized, it will not be possible to give confident answers. However, even with such knowledge and the ability to record from cells receiving natural inputs, we will also need models and methods which can scale beyond simple stimulation protocols and help us to distill from the data to what degree it is sensible to speak of particular computational subunits. To this end, we hope the hLN model (and future extensions of it) may prove helpful.

References

Bach, F., Jenatton, R., Mairal, J., & Obozinski, G. (2012). *Optimization with sparsity-inducing penalties.* Foundations and Trends in Machine Learning, 4(1):1-106.

Branco, T., Clark, B.A., & Häusser, M. (2010). *Dendritic Discrimination of Temporal Input Sequences in Cortical Neurons.* Science, 329(5999), 1671–1675.

Chen, X., Leischner, U., Rochefort, N. L., Nelken, I., & Konnerth, A. (2011). *Functional mapping of single spines in cortical neurons in vivo.* Nature, 475(7357), 501–505.

Dayan, P. and Abbott, L.F. (2001) *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems.* MIT Press.

Diba, K., Koch, C., & Segev, I. (2006). *Spike propagation in dendrites with stochastic ion channels.* Journal of Computational Neuroscience, 20(1), 77–84.

Duda, R.O., Hart, P.E., and Stork, D.G. (2001). *Pattern Classification.* Wiley-Interscience, 2nd edition.

Freeman, J., Field, G., Li, P.H., Greschner, M., Jepson, L.H., Rabinowitz, N.C., Pnevmatikakis, E., Gunning, D.E., Mathieson, K., Litke, A.M., Chichilnisky, E.J., Paninski, L., & Simoncelli, E.P. (2012). *Characterizing the nonlinear subunits of primate retinal ganglion cells.* Society for Neuroscience (SfN).

Gasparini, S. & Magee, J.C. (2006). *State-Dependent Dendritic Computation in Hippocampal CA1 Pyramidal Neurons.* Journal of Neuroscience, 26(7), 2088–2100.

Gerstner, W. & Kistler, W.M. (2002). *Spiking Neuron Models: Single Neurons, Populations, Plasticity.* Cambridge University Press.

Griffiths, T.L. & Ghahramani, Z. (2011). *The Indian buffet process: an introduction and review.* Journal of Machine Learning Research (JMLR), 12(April), 1185–1224.

Heller, K.A. & Ghahramani, Z. (2005). *Bayesian hierarchical clustering.* Proceedings of the 22nd International Conference on Machine Learning (ICML), 297–304.

Hines, M. & Carnevale, T. (1997). *The NEURON simulation environment.* Neural Computation 9:1179-1209.

Houlsby, N., Huszár, F., Ghahramani, Z., & Lengyel, M. (2011). *Bayesian Active Learning for Classification and Preference Learning.* Arxiv preprint arXiv:1112.5745.

Huggins, J. & Paninski, L. (2012). *Optimal experimental design for sampling voltage on dendritic trees.* Journal of Computational Neuroscience, 32: 347-66.

Huys Q.J.M., Ahrens M.B., & Paninski L. (2006). *Efficient estimation of detailed single neuron models.* Journal of Neurophysiology 96: 872-890.

Izhikevich, E.M. (2003). *Simple Model of Spiking Neurons.* IEEE Transactions on Neural Networks, 14:1569-1572.

Jia, H., Rochefort, N. L., Chen, X., & Konnerth, A. (2010). *Dendritic organization of sensory input to cortical neurons in vivo.* Nature, 464(7293), 1307–1312.

Knowles, D.A. & Ghahramani, Z. (2011). *Pitman-yor diffusion trees.* 27th Conference on Uncertainty in Artificial Intelligence (UAI).

London, M. & Häusser, M. (2005). *Dendritic computation.* Annual Review of Neuroscience, 28, 503–532.

Losonczy, A. & Magee, J.C. (2006). *Integrative Properties of Radial Oblique Dendrites in Hippocampal CA1 Pyramidal Neurons.* Neuron, 50(2), 291–307.

Macke, J.H., Cunningham, J.P., Byron, M.Y., Shenoy, K.V., & Sahani, M. (2011). *Empirical models of spiking in neural populations.* Advances in Neural Information Processing System (NIPS).

Magee, J.C. (2000). *Dendritic integration of excitatory synaptic input.* Nature Reviews Neuroscience, 1(3), 181–190.

Paninski, L. (2004). *Maximum likelihood estimation of cascade point-process neural encoding models.* Network: Computation in Neural Systems, 15(4), 243–262.

Pillow, J.W. (2007). *Likelihood-based approaches to modeling the neural code.* Bayesian brain: Probabilistic approaches to neural coding, 53–70.

Pillow, J.W., Shlens, J., Paninski, L., Sher, A., Litke, A.M., Chichilnisky, E.J., & Simoncelli, E.P. (2008). *Spatio-temporal correlations and visual signalling in a complete neuronal population.* Nature, 454(7207), 995–999.

Poirazi, P., Brannon, T., & Mel, B.W. (2003). *Pyramidal neuron as two-layer neural network.* Neuron, 37(6), 989–999.

Poirazi, P., Brannon, T., & Mel, B.W. (2003). *Arithmetic of subthreshold synaptic summation in a model CA1 pyramidal cell.* Neuron, 37(6), 977–987.

Polsky, A., Mel, B.W., & Schiller, J. (2004). *Computational subunits in thin dendrites of pyramidal cells.* Nature Neuroscience, 7(6), 621–627.

Schiller, J., Major, G., Koester, H.J., & Schiller, Y. (2000). *NMDA spikes in basal dendrites of cortical pyramidal neurons.* Nature, 404(6775), 285–289.

Schwartz, O., Pillow, J.W., Rust, N.C., & Simoncelli, E.P. (2006). *Spike-triggered neural characterization.* Journal of Vision, 6(4), 13–13.

Spruston, N. (2008). *Pyramidal neurons: dendritic structure and synaptic integration.* Nature Reviews Neuroscience, 9(3), 206–221.

Truccolo, W., Eden, U.T., Fellows, M.R., Donoghue, J.P., & Brown, E.N. (2005). *A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects.* Journal of Neurophysiology, 93(2), 1074–1089.

Ujfalussy, B. & Lengyel, M. (2011). *Active dendrites: adaptation to spike-based communication.* Advances in Neural Information Processing Systems (NIPS).

Varga, Z., Jia, H., Sakmann, B., & Konnerth, A. (2011). *Dendritic coding of multiple sensory inputs in single cortical neurons in vivo.* Proceedings of the National Academy of Sciences of the United States of America (PNAS), 108(37), 15420–15425.

Vintch, B., Zaharia, A., Movshon, J.A., & Simoncelli E.P. (2012). *Fitting receptive fields in V1 and V2 as linear combinations of nonlinear subunits.* Computational and Systems Neuroscience (CoSyNe), Feb 2012.

Zador, A. (2000). *The basic unit of computation.* Nature Neuroscience, 3 Suppl: 1167.