Attractor dynamics in rate-distortion networks trained via spike-timing dependent plasticity

Clayton Seitz

May 13, 2021

Table of contents

- Introduction
- 2 Channel coding for neural networks
- 3 Supervised training of low-rate critical networks
- Multivariate information theory
- 5 Adaptation of the transfer function
- 6 Learning an energy function over phase space
- Generalization bounds and density estimation
- The energy function defines a dynamical system
- 9 The energy function is a generative model
- Application to natural image statistics

Introduction

How can a network of neurons learn to categorize its inputs from a simple learning rule based on synaptic plasticity?

How can a network form a memory and how do later inputs retrieve that memory?

It may be that repeated stimulation leaves an "impression" on the network via plasticity rules and that impression is a kind of dynamical attractor

Channel coding for neural networks

Networks of neurons can be viewed as a communication channel Except this communication channel *learns* the transformation F based on the statistical structure of its input X. Visual cortex has learned an encoding for visual scenes (that perhaps maximizes information)

Leaky integrate and fire neurons

A realistic LIF model might look like

$$\tau_m \frac{dV[I]}{dt} = (V[I] - E) \sum_j W^0[I, j] + (V[I] - E_{in}) \sum_k W^1[I, k])$$

Instead, we ignore changes in the voltage of the postsynaptic neuron due to subthreshold voltages of the presynaptic neuron and let matrices W learn the input-output voltage relationship

$$V[j, t+1] = \alpha V[j, t+1] + \sum_{i \neq j} W_{ij}^{0} z[i, t] + \sum_{i} W_{ij}^{1} x[i, t+1] - z[j, t] v_{th}$$

where
$$z = H(v - v_{th})$$

RNN Gradients

Say we have a model $\Phi = (W^0, W^1)$ and want to use gradient descent to train a network to have a target rate or a target branching parameter. The rate and its associated loss for a single unit is

$$r(t) = rac{1}{\Delta t} \int_{t}^{t+\Delta t} d au \langle
ho(au)
angle \quad \mathcal{L} = lpha(r-r_0)^2$$

We would like the standard update

$$\Delta W_{ij} = -\eta \frac{\partial \mathcal{L}}{\partial W_{ij}}$$

But it is intractable to compute $\frac{\partial \mathcal{L}}{\partial W_{ij}}$ since $\rho(t)$ depends on other neurons through space and time.

Factorizing loss gradients for BPTT

BPTT involves unrolling an RNN into a large feedforward network where each layer is a time step.

$$\frac{\partial \mathcal{L}}{\partial W_{ij}^t} = \frac{\partial \mathcal{L}}{\partial h_j^t} \frac{\partial h_j^t}{\partial W_{ij}^t}$$

and the total gradient is a sum over the layers (time)

$$\frac{\partial \mathcal{L}}{\partial W_{ij}^t} = \sum_{t} \frac{\partial \mathcal{L}}{\partial h_j^t} \frac{\partial h_j^t}{\partial W_{ij}^t}$$

Deriving e-prop from BPTT

Consider the first term above. The hidden state is computed by some function $h_j^t = F(z_j^t, h_j^{t-1}, W)$. Backpropagating through time is then

$$\frac{\partial \mathcal{L}}{\partial h_j^t} = \frac{\partial \mathcal{L}}{\partial z_j^t} \frac{\partial z_j^t}{\partial h_j^t} + \frac{\partial \mathcal{L}}{\partial h_j^{t+1}} \frac{\partial h_j^{t+1}}{\partial h_j^t}$$

which must be expressed recursively

$$\begin{split} \frac{\partial \mathcal{L}}{\partial h_{j}^{t}} &= \frac{\partial \mathcal{L}}{\partial z_{j}^{t}} \frac{\partial z_{j}^{t}}{\partial h_{j}^{t}} + \left(\frac{\partial \mathcal{L}}{\partial z_{j}^{t+1}} \frac{\partial z_{j}^{t+1}}{\partial h_{j}^{t+1}} + (...) \frac{\partial h_{j}^{t+2}}{\partial h_{j}^{t+1}} \right) \frac{\partial h_{j}^{t+1}}{\partial h_{j}^{t}} \\ &= \mathcal{L}_{j}^{t} \frac{\partial z_{j}^{t}}{\partial h_{j}^{t}} + \left(\mathcal{L}_{j}^{t+1} \frac{\partial z_{j}^{t+1}}{\partial h_{j}^{t+1}} + (...) \frac{\partial h_{j}^{t+2}}{\partial h_{j}^{t+1}} \right) \frac{\partial h_{j}^{t+1}}{\partial h_{j}^{t}} \\ &= \mathcal{L}_{j}^{t} \frac{\partial z_{j}^{t}}{\partial h_{j}^{t}} + \left(\mathcal{L}_{j}^{t+1} \frac{\partial z_{j}^{t+1}}{\partial h_{j}^{t+1}} + (...) \frac{\partial h_{j}^{t+2}}{\partial h_{j}^{t+1}} \right) \frac{\partial h_{j}^{t+1}}{\partial h_{j}^{t}} \end{split}$$

Deriving e-prop from BPTT

Plugging into the original factorization gives

$$\frac{\partial \mathcal{L}}{\partial W_{ij}} = \left(\sum_{t} L_{j}^{t} \frac{\partial z_{j}^{t}}{\partial h_{j}^{t}} + \left(L_{j}^{t+1} \frac{\partial z_{j}^{t+1}}{\partial h_{j}^{t+1}} + (...) \frac{\partial h_{j}^{t+2}}{\partial h_{j}^{t+1}}\right) \frac{\partial h_{j}^{t+1}}{\partial h_{j}^{t}}\right) \frac{\partial h_{j}^{t'}}{\partial W_{ij}}$$

You can then collect terms that are multiplied \mathcal{L}_{j}^{t}

$$\frac{\partial \mathcal{L}}{\partial W_{ij}} = \sum_{t} L_{j}^{t} \frac{\partial z_{j}^{t}}{\partial h_{j}^{t}} \left(\sum_{t' \leq t} \left(\prod_{t'} \frac{\partial h_{j}^{t'+1}}{\partial h_{j}^{t'}} \right) \frac{\partial h_{j}^{t'}}{\partial W_{ij}} \right)$$
$$= \sum_{t} L_{j}^{t} \frac{\partial z_{j}^{t}}{\partial h_{j}^{t}} \epsilon_{ij}^{t} = \sum_{t} L_{j}^{t} e_{ij}^{t}$$

The eligibility vector for LIF neurons

We would like to know the eligibility is per synapse ϵ_{ij}^t for a network of RNN neurons.

Gradients can be computed using e-prop if we use a pseudo-derivative $\psi_j^t = \frac{\partial z_j^t}{\partial v_j^t}$ and use the fact that the eligibility vector ϵ is just a low passed filter of z.

$$\Delta W_{ij} = -\eta \sum_{t} \frac{\partial \mathcal{L}}{\partial z_{j}^{t}} \psi_{j}^{t} \mathcal{F}_{\alpha}(z_{i}^{t})$$

Constraining the global firing rate distribution

We can define a constraint on the variance of the global firing rate (which simultaneously constrains the mean)

$$\mathcal{L} = \beta(\sigma - \sigma_r)^2$$
 $\sigma = \frac{1}{T} \sum_{r} (r - \mu_r)^2$

where we constrain branching by constraining the variance s of the global firing rate where branching $\to 1$ as $s \to 0$.

$$L_j^t = \frac{\partial \mathcal{L}}{\partial z_j^t} = \frac{\partial \mathcal{L}}{\partial \sigma} \frac{\partial \sigma}{\partial n} \frac{\partial n}{\partial z_j^t} = \pm \beta (\sigma - \sigma_r) \cdot (r - \mu_r)$$

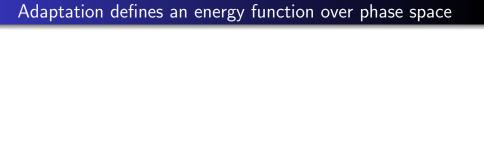
Think push-pull. Some variation is necessary for refractoriness.





Adaptation of the transfer function

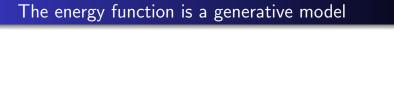
How do neuron transfer functions adapt to stimuli in an unsupervised manner?



Generalization bounds

What is the distance of a code defined by a particular energy function ${\sf E}$

The energy function defines a dynamical system



Application to natural image statistics