Training a Network of Spiking Neurons with Equilibrium Propagation

Peter O'Connor, Efstratios Gavves, Max Welling
QUVA Lab
University of Amsterdam

Motivation

- Backpropagation is biologically implausible for several reasons (Crick, 1989), two of which are:
  1. Biological neurons don't appear have a mechanism to propagate gradients backwards across synapses.
  2. Backprop involves neurons communicating continuous values, whereas biological neurons send binary "spikes".
- Recently, (Scellier and Bengio, 2017) proposed Equilibrium Propagation, which shows how neural networks might achieve gradient descent despite lacking a backward-signalling mechanism, addressing problem (1)

We propose how we might still use Equilibrium propagation in a setting where neurons are constrained to only communicate binary values, addressing problem (2).

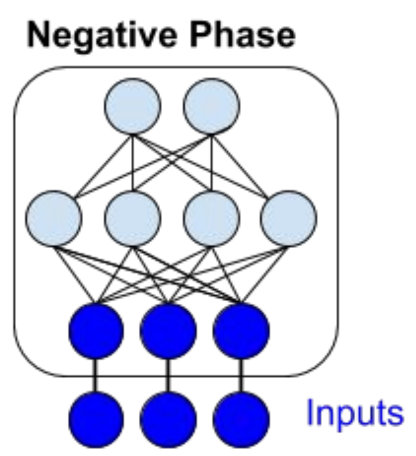
Long term vision: A scalable design for neural computing hardware - Neurons are physically implemented on a chip, have rich internal dynamics but are loosely coupled - running asynchronously and communicating with low-bandwidth bitstreams.

Equilibrium Propagation

(Scellier & Bengio, 2017) propose how one can train neural networks with a simple "no backprop" interface. Train a "continuous hopfield model" whose dynamics follow an energy function:

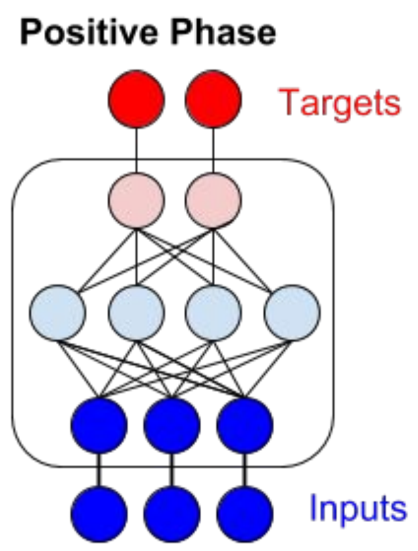
$$\frac{\partial s}{\partial t} \propto \frac{\partial E}{\partial s}$$

Negative Phase: Clamp input units to data, allow network to settle to fixed-point s^- of energy function:



$$E(s) = \frac{1}{2}\sum_u s_i^2 - \sum_{i \neq j} w_{ij}\rho(s_i)\rho(s_j) - \sum_i b_i \rho(s_i)$$

Positive Phase: Weakly clamp output units to target, move towards new fixed point s^+ of "perturbed" energy function:



$$E^\beta = E(s) + \beta C(s_{out}, y)$$

Update: Update based on contrastive loss between two fixed points. This minimizes output loss:

$$\Delta\theta \propto \left(\frac{\partial E^\beta(s^+)}{\partial \theta} - \frac{\partial E(s^-)}{\partial \theta}\right) \propto \frac{\partial C(s_{out}, y)}{\partial \theta}$$

Quantizing Neurons

Now we want to quantize inter-neuron communication.

At each step in dynamics, neuron may produce 0 or 1.

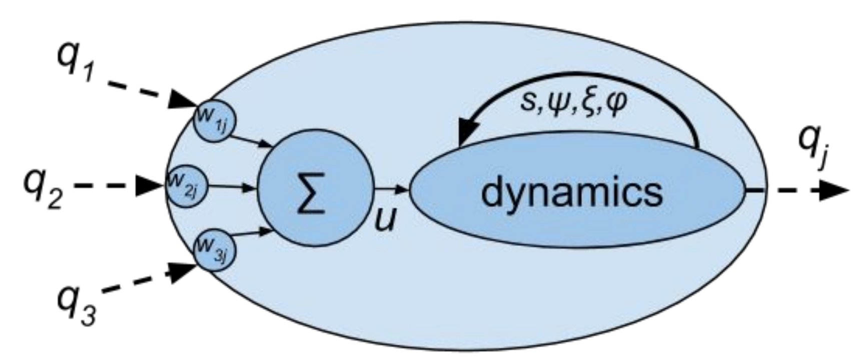
We want to converge to the same fixed-point as the real-valued network.

Continuous Valued Network

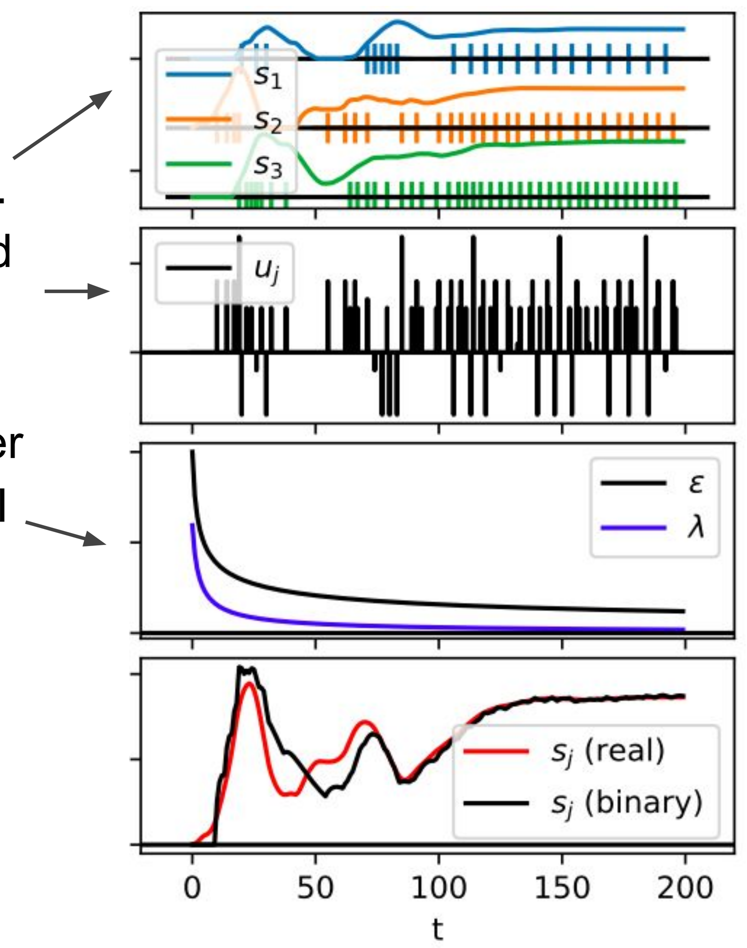
$$s_j^t = \left[(1-\epsilon)s_j^{t-1} + \epsilon\rho'(s_j^{t-1})\left(\sum_i w_{ij}\rho(s_i^{t-1}) + b_j\right)\right]_0^1$$

Binary-Valued Network

$$u_j^t = \sum_i w_{ij} q_i^{t-1}$$
$$v_j^t, \psi_j^t = dec(u_j^t, \psi_j^{t-1})$$
$$\epsilon_j^t, \xi_j^t = anneal(\epsilon_j^{t-1}, v_j^t, \xi_j^{t-1})$$
$$s_j^t = [(1-\epsilon_j^t)s_j^{t-1} + \epsilon_j^t \rho'(s_j^{t-1})(v_j^t + b_j)]_0^1$$
$$q_j^t, \phi_j^t = enc(\rho(s_j^t), \phi_j^{t-1})$$



- Neuron receives input from presynaptic neurons which quantize their activations.
- A neuron sees only a stream of weighted inputs.
- Early in convergence - inputs nonstationary, need high-step size. Later in convergence, inputs stationary but still noisy. Need annealing step-size to average out noise.



Defining the Encoding Scheme

(1) Naive Approach
Neurons stochastically represent their values, and integrate them with an annealing step-size:

$$q^t = \mathrm{Bern}(\rho(s^t)) \quad \text{Stochastic Encoder}$$
$$v^t = u^t \quad \text{Identity Decoder}$$
$$\epsilon^t = \frac{1}{(t)^\eta} \quad \text{Annealer}$$

Where $\eta \in (½,1)$ defined the annealing rate

(2) Faster Convergence with Sigma-Delta Modulation
Instead of stochastically representing values, we can converge faster with a stateful encoder:

$$\phi' = \phi^{t-1} + x^t$$
$$q^t = \left[\phi' > \frac{1}{2}\right] \quad \text{Sigma Delta Encoder}$$
$$\phi^t = \phi' - q^t$$

(3) Predictive Coding
Use bits to communicate *changes* in signal, reconstruct signal on receiving end.

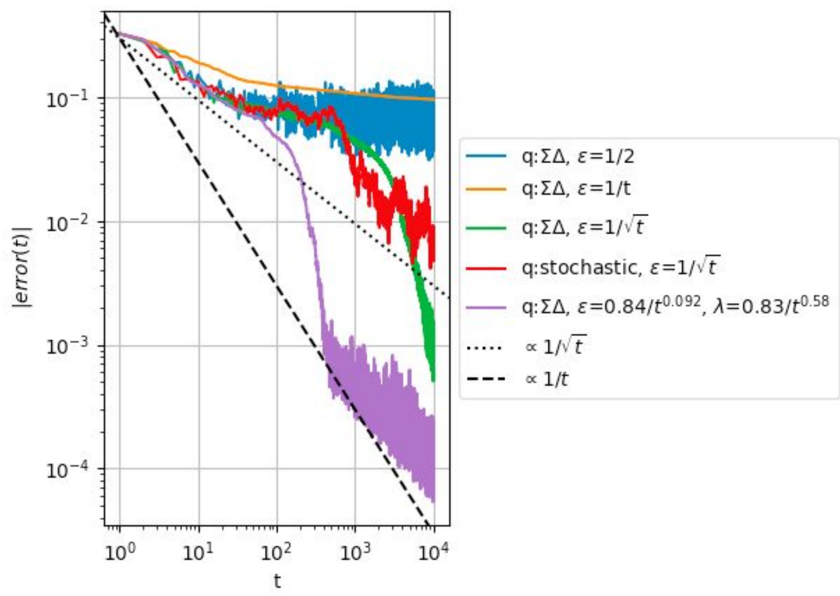
$$a^t = \frac{1}{\lambda}(\rho(s^t) - (1-\lambda)\rho(s^{t-1})) \quad \text{Predictive Encoder}$$
$$q^t = Q(a^t)$$
$$u_j^t = \sum_i w_{ij} q_i^{t-1}$$
$$v_j^t = (1-\lambda)v_j^{t-1} + \lambda u_j^t \quad \text{Predictive Decoder}$$

Where $\lambda \in (0, 1)$ is the degree to which bits represent *increments* to signal value. Like ϵ, λ can be annealed.

Experiments

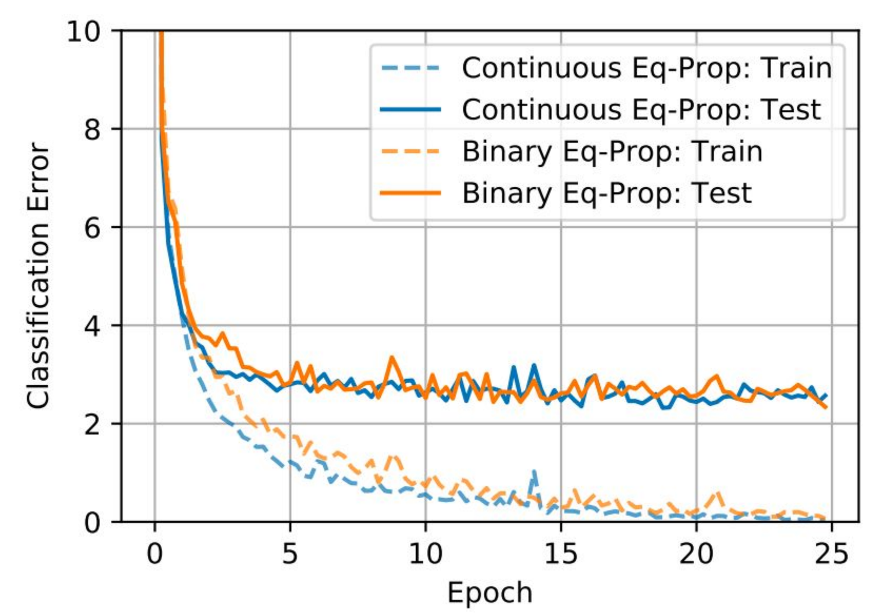
Convergence on Randomly Initialized Network

- Randomly initialize network,
- Compare convergence rates under different settings under random input.
- Do random search in annealing parameters for ϵ, λ
- We find that the best scheme involves annealing the prediction coefficient λ



Binary Eq.Prop on MNIST

- Use best converging scheme to train using Equilibrium Prop on MNIST.
- Comparable results to continuous-valued network.
- Best-performing encoding scheme performs similarly to continuous equilibrium prop.



References

1. Francis Crick. The recent excitement about neural networks. Nature, 337(6203):129–132, 1989
2. Benjamin Scellier and Yoshua Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. Frontiers in computational neuroscience, 11:24, 2017.