View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by University of Waterloo's Institutional Repository

Bidirectional Learning in Recurrent Neural Networks Using Equilibrium Propagation

by

Ahmed Faraz Khan

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2018

© Ahmed Faraz Khan 2018

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Neurobiologically-plausible learning algorithms for recurrent neural networks that can perform supervised learning are a neglected area of study. Equilibrium propagation is a recent synthesis of several ideas in biological and artificial neural network research that uses a continuous-time, energy-based neural model with a local learning rule. However, despite dealing with recurrent networks, equilibrium propagation has only been applied to discriminative categorization tasks. This thesis generalizes equilibrium propagation to bidirectional learning with asymmetric weights. Simultaneously learning the discriminative as well as generative transformations for a set of data points and their corresponding category labels, bidirectional equilibrium propagation utilizes recurrence and weight asymmetry to share related but non-identical representations within the network. Experiments on an artificial dataset demonstrate the ability to learn both transformations, as well as the ability for asymmetric-weight networks to generalize their discriminative training to the untrained generative task.

Acknowledgements

Firstly, I would like to express my gratitude for my supervisor, Prof. Jeff Orchard, for his continuous support and for guiding this thesis in the right direction, and my labmate Ehsan Ganjidoost, who has witnessed my spiral into a severe caffeine dependency over the past 2 years. I also appreciate my thesis readers, Profs. Justin Wan and Yaoliang Yu, for taking the time to review this work.

I gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

Finally, I would like to thank my family and friends for their support and encouragement, and the fine establishment of Burrito Boyz at 255 King Street North, Waterloo, for fueling the writing of this thesis.

Table of Contents

List of Figures vii

1 Introduction 1

2 Background 6

2.1 Biological Plausibility . . . 6

2.2 Backpropagation and Weight Symmetry . . . 8

2.3 Local Learning Rules . . . 12

2.4 Error Functions and Energy-Based Models . . . 13

2.5 Contrastive Learning . . . 17

2.6 Equilibrium Propagation . . . 19

2.7 Timescale and Gradient Dynamics . . . 22

2.8 Generative Models and Recurrence . . . 23

3 Bidirectional Equilibrium Propagation 26

3.1 Network Architecture . . . 26

3.1.1 Energy Dynamics . . . 27

3.2 Discretized Simulation . . . 28

3.3 Dataset . . . 30

3.3.1 Learning Algorithm . . . 31

4 Results and Discussion **33**

4.1 Results . . . 33

4.1.1 Bidirectional Equilibrium Propagation . . . 34

4.1.2 Restricted Bidirectional Equilibrium Propagation . . . 34

4.2 Biological Plausibility . . . 37

5 Conclusion **40**

5.1 Future Work . . . 41

5.1.1 Bidirectional Learning . . . 41

5.1.2 Finite Input Clamping . . . 41

5.1.3 Time-Varying Environment . . . 42

5.1.4 Biological Realism . . . 42

5.1.5 Deep Networks . . . 43

References **44**

List of Figures

1.1 Artificial neuron with inputs 2

2.1 Feedback Alignment architectures 10
2.2 A Hopfield network 15
2.3 A Restricted Boltzmann Machine 17
2.4 Impulse response of recurrent architectures 25

3.1 Bidirectional Equilibrium Propagation and its directed modes 29
3.2 Example dataset class archetypes 32

4.1 Bidirectional learning over 20 epochs, with tied weights 35
4.2 Bidirectional learning over 20 epochs, with asymmetric weights 36
4.3 Restricted bidirectional learning over 20 epochs, with asymmetric weights . 38

Chapter 1

Introduction

Artificial neural networks (ANNs) are a computing paradigm roughly based on the distributed network of neurons in the brain, with applications in machine learning. Typically, neurons are treated as nearly identical, applying a simple non-linear function to the sum of their inputs and passing the resulting signal to other neurons. While each neuron is simple, collections of such units are able to learn and implement complex transformations.

Mathematically, a neural network is a graph of homogeneous non-linear units, with directed and weighted synaptic connections between units as shown in Figure 1.1. These networks are typically, but not necessarily, organized into input and output layers, with one or more "hidden layers" sandwiched in between. Each unit (or neuron) applies a non-linear *activation function* to the weighted sum of its *pre-synaptic* inputs, and then outputs this signal (or "activity") to its *post-synpatic* neurons. Architecturally, ANNs can be characterized as either feedforward (where signals propagate from source to sink along a directed acyclic graph) or recurrent (where lateral connections or feedback loops of some sort exist).

Neural networks are turned into useful forms by "learning" desired input-output transformations via supervised methods, or by unsupervised extraction of features from inputs without target outputs. In either case, such transformations are parametrized by connection weights between neurons. Thus, learning constitutes modifying synaptic weights for a desired change in input-response. Since neural networks are typically high-dimensional systems, with each neuron influencing the eventual output, updating weights to minimize output error is a non-trivial multivariate optimization problem. The process of determining weight updates for some data is known as the *credit assignment problem* of neural networks.

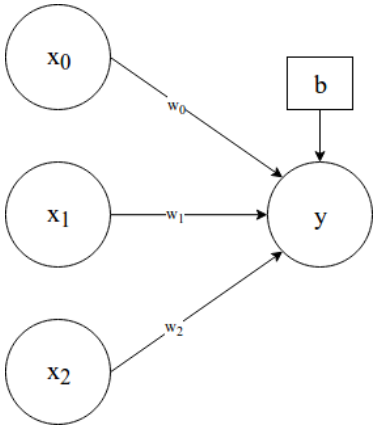


Figure 1.1: An artificial neuron with inputs and a bias, implementing $y = f(\sum_i w_i x_i + b)$ for weights w_i, bias b, pre-synaptic input neurons x_i and post-synaptic neuron y with activation function f.

A feedforward network can be organized such that each neuron belongs to exactly one hierarchical layer. Neural activity begins at the input layer and experiences a series of non-linear transformations. Modifying a synaptic weight affects only higher layers, leading to a simpler optimization problem than recurrent neural networks. Thus, solutions to the feedforward credit assignment exist, primarily based on the backpropagation algorithm [25] using the chain rule of calculus on the gradient of output error. These update rules allow feedforward networks to generalize a variety of input-output transformations from training data examples. It has been shown that a feedforward network with at least one hidden layer, and a continuous, non-constant and bounded activation function is able to learn an arbitrary input-output function, given enough hidden layer units between input and output layers [20].

However, recurrent neural networks have proven more difficult to tame, due to the complexity of the credit assignment problem in the presence of cycles. In a feedforward network, the activities of neurons $\{x_i^{l+1}\}$ in layer $l+1$ are conditionally independent of neurons $\{x_i^{l-1}\}$ in layer $l-1$, given the activities $\{x_i^l\}$. In general, such a claim cannot be made for a recurrent neural network. The consequences of changes to connection weights are not straightforward to predict, whereas backpropagation guarantees a reduction in output error given a small enough update step size. Recurrent connections open the Pandora's box of dynamical systems behaviour, including divergent activity and sensitivity to initial states.

Biologically, however, connections between brain areas are often reciprocal, and the

resulting network effects of this recurrent connectivity can influence function [12]. Phenomena such as working memory are difficult to explain without recurrent models [8]. Similarly, recurrent neural networks are better suited to certain (artificial) applications involving dynamic memory, such as audio/speech, language/text, and other forms of time series data. Thus, a convincing biological neural network model must handle recurrent connections. While some efforts have been made to adapt backpropagation to recurrent neural networks [44], these methods tend to not scale well as RNNs are temporally unfurled. In addition, from the perspective of neuroscience, backpropagation is considered biologically implausible (Section 2.2).

While neural learning algorithms for recurrent networks do exist, they are generally a less satisfactorily solved problem. Backpropagation-through-time, a variant applied to an unrolled recurrent network, suffers from the compromise of temporal truncation, and is only really applicable to a finite impulse network. Additionally, backpropagation-through-time is particularly susceptible to vanishing and exploding gradients (discussed in Sec 2.2) [33]. Other algorithms generally show limited consideration of biological plausibility (with the exception of Hebbian-like rules discussed in Section 2.3). Contrastive methods, which are the most promising basis for the models considered in this thesis, lack a guarantee of convergence [42].

Research in the propagation of activities through a network of simple neuronal units comes from two fields with slightly different foci. Machine learning is primarily concerned about statistical performance in terms of modeling training data, while computational neuroscience is interested in neural networks with the additional constraint that they are consistent with biological observations.

To contextualize the neuroscientific interest in artificial neural networks and the desired properties of such models, it is useful to consider the level of modeling abstraction. Neurobiological models of a process, such as control of a limb, are often described in their scope in terms of Marr's levels of analysis [6].

- *Computational models* state the problem to be solved, without constraining the method of their solution. This may be a kinematic model of the limb that the motor cortex articulates, with desired joint states or trajectories.
- *Algorithmic models* describe how problems may be solved, agnostic to how this may be realized. This may be a general description of the type of error-feedback used to generate motor commands.
- *Implementational models* describe the exact mechanism of the solution. At the extreme end, this would be a neuromorphic spiking neural network, whose architecture

and hyperparameters are informed by biology, that implements the algorithmic objective with close correspondence to biological data.

For our purposes, each level is built on its preceding one. In the context of a neurobiological process, implementational models represent a more "complete" understanding of a neural process, and are thus more difficult to achieve in synergy with an algorithmic solution to a computational task. That is, we do not consider a detailed, biologically-realistic neuron model, with sophisticated simulations of neuronal electrical properties, in itself to be an implementational level model because it does not implement any algorithmic solution to a computational problem faced by a neurobiological system. Likewise, we do not consider neuroscientific, experimentally-derived learning rules such as spike timing-dependent plasticity (STDP) [40] in isolation, but only in the context of network-level task performance.

However, if the difficult task of integrating this low-level model with a higher level task and algorithm is achieved, it would classify as an implementational level model. In addition, particular models may fall between levels on this spectrum. Typical neural networks models lie between algorithmic and implementational levels, being more "neural-like" than an algorithmic description, but falling short of an implementational replica.

Typically, the network-level research is dominated by artificial neural networks and machine learning. One learning framework that has attempted to bridge the gap between single-neuron models and networks, as well as pushing towards an implementational Marr levels, is equilibrium propagation (EP) [37]. Equilibrium propagation is an artificial neural network model, but one that is intended to incorporate more biologically realistic neuron models, network architectures and synaptic plasticity (i.e. weight updates) without sacrificing high-level task performance.

Equilibrium propagation uses only local computations, with signals physically accessible to a neuron, with a general Hebbian weight update algorithm applicable to recurrent networks. It has been shown that equilibrium propagation allows neural networks with 1-3 hidden layers to learn to classify images of the benchmark MNIST dataset of digits with test accuracies approaching the state of the art [37].

This thesis introduces bidirectional equilibrium propagation (BEP), which extends equilibrium propagation from the classification task to the generative one. While the classification task involves mapping a D-dimensional input vector $\mathbf{x}$ to its correct class y by learning the discriminative model distribution $P(Y|X = \mathbf{x})$, the generative task performs the complimentary action of producing a sample given the class by learning the joint probability distribution $P(X, Y)$ in the same network as well.

Conventionally, in neuroscience, the sensitivity of neurons to different inputs is characterized by their *receptive fields*, which is the region of the input data and the features in that region that most excite a neuron. Similarly, feedforward artificial networks, such as convolutional networks [47], interpret the functional roles of neurons, parametrized by their preceding connection weights, as filters. For example, in a visual neural network, a neuron in a certain layer that has high values when two lines intersect at a particular spatial location is interpreted as a corner detector filter.

While this is a useful characterization in purely feedforward networks, it is an incomplete description of a neuron's role in a network with recurrent connections. In such a case, the network becomes a dynamical system, and is perhaps better described in terms of attractor states of neural activity rather than a stacked hierarchy of static neural responses. Furthermore, real biological neurons are involved in multiple tasks. Sensory processing, for example, is really an active process, receiving feedback from higher area neurons which guides stimulus selection and attention [39]. Even early-stage auditory cortex neurons respond to the same stimuli differently, depending on the behavioural context [11]. Friston and Price [10] also argue that what the activity of a neuron represents is dynamic (i.e. time-dependent) and contextual (i.e. input-dependent).

The bidirectional equilibrium propagation model in this thesis is continuous in time. Temporal dynamics occur on two time scales: the fast, transient energy-minimization dynamics of neuronal states, and the slower settling of the network into a steady state equilibrium. Additionally, a distinction must be made between both of these scales of *activity* dynamics and the synaptic plasticity that occurs due to the contrastive equilibrium propagation learning rule (described in Section 2.6).

Bidirectional equilibrium propagation attempts to reproduce a type of task multiplicity: invertible representations. By learning the weights that satisfy both a bottom-up, discriminative task, as well as a top-down, generative task, neurons are involved in distinct computations that share the same parameters (i.e. connection weights), but a different network energy.

Chapter 2

Background

Bidirectional equilibrium propagation is composite model synthesizing several ideas from neuroscience and artificial neural networks. This section summarizes the influences of the model, from the research objective of biological plausibility to bidirectional and asymmetric weights, energy-based learning, contrastive methods and generative models.

2.1 Biological Plausibility

The motivation to study biologically-plausible algorithms derives in part from potential computation modeling contributions to neuroscience, as well as from the potential of architectural or algorithmic contributions to artificial neural networks. While artificial neural networks were first inspired by biology, the basic neuron model used has not changed much since the early perceptron [35] in the 1950s. Meanwhile, several decades of experimental research on the complex mechanisms of biological neural signal processing has been neglected. A biologically-realistic neural network model that is also functionally capable of machine learning task performance would thus serve both fields.

Broadly, the biological plausibility of a network model is evaluated by its adherence to known neurobiological principles and observations. In practical terms, this does not necessarily mean perfect fidelity, and some criteria are more critical to biological realism than others. For example, while biological neurons transmit action potentials (rapid voltage spikes) when their cell membrane potential exceeds a certain threshold, most artificial models transmit continuously-valued activities. These activities are interpreted as some running average of the spiking rate, and so these abstractions are called rate neurons.

While many neuromorphic networks do use spiking neurons, particularly to try to learn and use possible representations encoded in the relative timings of spikes [22], and the goal of biologically plausible learning rules is to handle the spiking neuron case, the rate neuron approximation is often forgiven. However, to claim that a group of neurons implements a learning algorithm that computes the product of two distant, non-adjacent neurons' activations would be less believable, as this global information is not available in a biological setting.

As a guideline, O'Reilly lists 6 interrelated criteria for biologically-plausible cortical models, divided into three principles [32]:

- **biological realism**: models should not violate known biological constraints (e.g. weight-copying)
- **architectures** reflect the structure and functional organization of biological networks
  - *distributed representations*: encoding information among a population of simple units, rather than in a centralized memory
  - *inhibitory competition*: allowing only the strongest activations to prevail for sparse representations (similar to max-pooling in a convolutional neural network)
  - *bidirectional activation propagation*: higher areas influencing lower ones using recurrence and feedback to resolve ambiguity
- **learning** is split into task-specific and general environmental components
  - *error-driven task learning*: using supervised learning error signals
  - *Hebbian model learning*: collects correlated environmental stimulus statistics in an unsupervised way

O'Reilly emphasizes the interactions between these principles, and how each may resolve problems introduced by the others. For instance, distributed representations and inhibitory competition strike a balance between competition and cooperation, or between a winner-takes-all level of sparsity and conditionally-independent neuron activities. O'Reilly argues that approaches that successfully integrate multiple biological principles may be more robust than, for example, manually added extra parameters to enforce criteria such as sparsity, namely in the case of bidirectional networks where activity competition can provide a form of dynamic thresholding to restrain runaway positive feedback.

2.2 Backpropagation and Weight Symmetry

Backpropagation is a supervised learning algorithm based on gradient descent on the cost or error function E [36]. This error function is a function of the state of the output layer and some target output data. The credit assignment problem is solved by using the chain rule to assign the "blame" for the output error to each neuron, working backwards from the output layer. This gives the partial derivative of the error function with respect to the weight $w^l_{i,j}$

$$\frac{\partial E}{\partial w^l_{ij}} = \frac{\partial E}{\partial a_j}\frac{\partial a_j}{\partial s_j}\frac{\partial s_j}{\partial w_{ij}}, \tag{2.1}$$

where the weighted sum of inputs is $s = \sum_k (w^l_{kj} a^{l-1}_k)$, and a^l_k is the activity of the k^{th} neuron of layer l.

This informs the direction of the weight update

$$\Delta w^l_{ij} = -\gamma \frac{\partial E}{\partial w^l_{ij}} \tag{2.2}$$

where the learning rate is γ and w^l_{ij} is the weight from pre-synaptic neuron i of layer $l-1$ to post-synaptic neuron j of layer l.

However, backpropagation faces issues in terms of biological plausibility [41, 4]. Networks must operate in distinct forward and backward modes. The latter, during which the errors are back-propagated, is linear, in contrast to the nonlinear operation of biological neurons [23]. In addition, evaluating the weight updates during the backpropagation phase requires computing the derivative of neuronal activation functions at its value during the forward phase. The existence of a regulatory mechanism that alternates between forward and backward passes is also contentious [41]. In addition, supervised learning methods in general face the issue of how neurons may have access to target outputs at all.

Another criticized aspect of backpropagation is the the linear backwards error propagation down the hierarchy as in Equation 2.1. However, biological neurons are non-linear, making them unlikely to perform this sort of computation.

Perhaps the most common criticism of the biological plausibility of backpropagation is its requirement of weight symmetry [4, 41]. Evidently, this back-propagating signal uses the same connection weights as the forward signal. As real biological synapses are unidirectional, this algorithm implies that there would be a separate, parallel network with the same (but transposed) weights running backwards to communicate weight updates. An

update to the feedforward network would require an instantaneous weight update to the backward network, so that further weight updates are appropriately handled by the backpropagation chain rule. This copying of weights is considered highly unlikely in biological substrates due to known physical and biological constraints. Thus, the dilemma posed by the requirement of the forward and backward networks to share identical weights w_{ij}^l is known as the weight-transport problem.

In recent years, however, several works have found that such a parallel network need not necessarily have identical weights. Lillicrap et al. used random and fixed weights for the feedback network [28], with comparable task performance with backpropagation. They note that backpropagation uses the error signal $W^T\mathbf{e}$, where W is the (forward) weight matrix (composed of the elements w_{ij} and $\mathbf{e}$ is the output error vector. In the backward network, the forward weights may be substituted by any random weight matrix B and learning would still occur, as long as the condition on B in Equation 2.3 is met.

$$\mathbf{e}^T W B \mathbf{e} > 0 \tag{2.3}$$

Intuitively, this requires that the feedback signals $B\mathbf{e}$ and $W^T\mathbf{e}$ point in the same direction (i.e. less than 90 degrees apart). Of course, a better alignment results in faster learning, which can be achieved even with a fixed B by adjusting only W. In a network with one hidden layer, Lillicrap et al. used standard backpropagation to adjust W, except for the first layer of connections W_0 from the input $\mathbf{x}$, which were adjusted based on the feedback alignment learning rule,

$$\Delta W_0 = \eta B \mathbf{e} \mathbf{x}^T \tag{2.4}$$

where η is some scalar learning rate.

Similarly, direct feedback alignment (DFA) and indirect feedback alignment (IFA) [31] achieved good results on the MNIST and CIFAR datasets using the output error projected back to each layer (depicted in Figure 2.1). It breaks the reciprocal error feedback of both backpropagation and Lillicrap et al.'s feedback alignment; that is, layers connected in a forward pass do not necessarily connect backwards in an error pathway. These architectures present alternatives to the standard feedforward structure that may be worth exploring.

Liao et al. [27] quantified the performance of 6 different levels of weight symmetry in asymmetric backpropagation, ranging from sign-concordant feedback (where forward and backward weights shared the same signs for each of their entries, but not magnitudes) to completely random but fixed feedback. They concluded that the magnitudes of the feedback weights do not affect performance too much, but the signs do matter. In fact, random

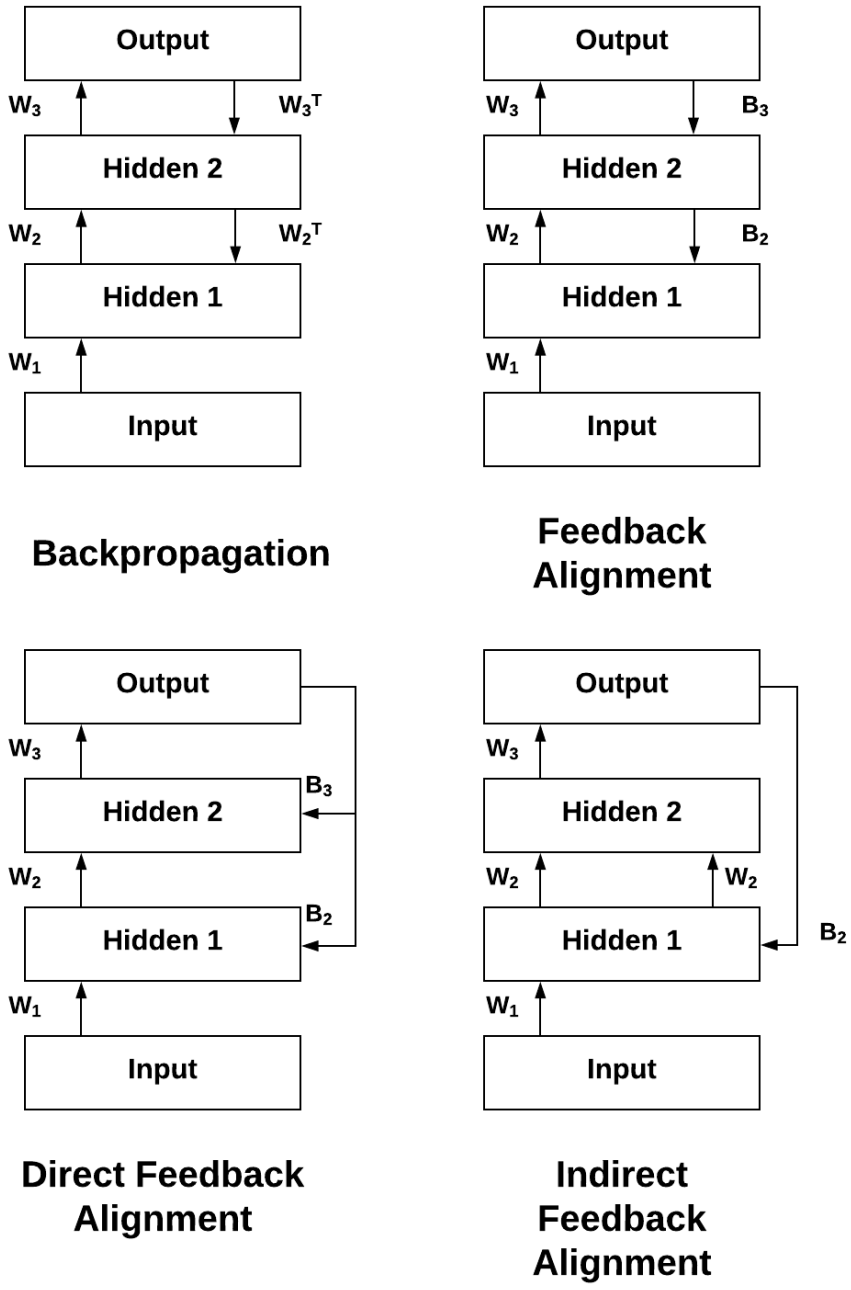


Figure 2.1: *Feedback Alignment:* Variations of the feedback alignment principle, compared with the standard backpropagation architecture. Error pathways are on the right of each diagram.

magnitudes with sign-concordance perform comparably to stochastic gradient descent, although some input-normalization is required for such methods to work.

Target propagation [26] takes a different approach. Rather than using the chain rule to transfer gradients of output error backwards across layers, target propagation assigns a target activity vector $\hat{\mathbf{h}}_l$ to each layer l by calculating an "approximate inverse" g_l such that, for all layers l,

$$f_i(g_i(\mathbf{h}_i)) \approx \mathbf{h}_i \quad \text{or} \quad g_i(f_i(\mathbf{h}_{i-1})) \approx \mathbf{h}_{i-1}. \tag{2.5}$$

Given these layer-wise inverses $\{g_i\}$, target propagation addresses the vanishing gradient problem, where deep networks with highly non-linear activation functions have back-propagated gradients that are very small or zero. Target propagation does not enforce the biologically implausible, linear computation nor symmetric backward weights of backpropagation. It does not require evaluation of the operating point of the feedforward activation function. The weights w_{ij}^l from pre-synaptic neuron i layer $l-1$ to the post-synaptic neuron j in layer l are then updated using stochastic gradient descent on a (local) layer error function

$$E_l(\mathbf{h_l}, \hat{\mathbf{h}}_l) = ||\mathbf{h}_l - \hat{\mathbf{h}}_l||^2 \tag{2.6}$$

$$\Delta w_{ij}^l = -\eta \frac{\partial E_l(\mathbf{h_l}, \hat{\mathbf{h}}_l)}{\partial w_{ij}^l} \tag{2.7}$$

where $\mathbf{h}_l$ is the actual activity vector of layer l and η is the learning rate.

The target for a preceding $i-1$ is given by

$$\hat{\mathbf{h}}_{i-1} = g_i(\hat{\mathbf{h}}_i). \tag{2.8}$$

Naturally, the base case of the output layer uses the target $\hat{\mathbf{h}}_L = \mathbf{y}$ (where $\mathbf{y}$ is the target output vector from the training data) is given by

$$\Delta w_{ij}^L = -\eta \frac{\partial E_L(\mathbf{h_L}, \mathbf{y})}{\partial w_{ij}^L}. \tag{2.9}$$

Thus, target propagation essentially amounts to learning layer-wise auto-encoders (i.e. networks that learn a sparse representation of the input in the hidden layer by training to reconstruct the input at the output layer), so that the backward weight update pass can be done using local computations only. While it performs comparably to backprop-trained networks [26], it, like backpropagation, does not address learning in recurrent neural

networks. It also trades the issue of weight symmetry for the problem of inverting each layer.

Backpropagation is the dominant learning algorithm, at least for feedforward neural networks. It can be extended to *unrollable* neural networks (whose recurrent cycles can be removed by temporally expanding the network as depicted in Figure 2.4a). Nevertheless, research into backpropagation has recently been the field for the study of asymmetric weights and alternative methods of credit assignment such as target propagation, factors that appear in bidirectional equilibrium propagation.

2.3 Local Learning Rules

An alternative to backpropagation and reciprocal error pathways is Hebbian learning, where connections are strengthened by the correlated firing of pre- and post-synaptic neurons (i.e. before and after the connection, or x and y in Equation 2.10, respectively). The standard Hebb's rule is typically stated as:

$$\Delta w_{ij} = \eta x_i x_j \tag{2.10}$$

This unsupervised form of synaptic learning requires no error signal and uses only local signals: pre- and post-synaptic activity. Consequently, it is often the basis of attempts at biologically-plausible models, whether in this form or as another, more stable variant (e.g. BCM rule, Oja's rule, Sanger's rule etc.).

Spike-timing-dependent plasticity (STDP) [29] is the dominant spiking variant of Hebbian learning. As an empirical model, it uses a window function $Q(t)$ (Equation 2.12) on the difference between the pre-synaptic spike arrival time t_i^k and the post-synaptic spiking time t_j^l to determine the potentiation or depression (i.e. strengthening or weakening, respectively) of a synapse with weight w_{ij} between neurons indexed i and j. For N pre-synaptic spike arrivals and consequently N post-synaptic firing events, the weight update is determined by

$$\Delta w_{ij} = \sum_{k=1}^{N} \sum_{l=1}^{N} W(t_j^l - t_i^k) \tag{2.11}$$

where the window function is

$$Q(t) = \begin{cases} A_+ \exp(-t/\tau_+) & \text{if } t > 0 \\ -A_- \exp(t/\tau_-) & \text{if } t < 0 \\ 0 & \text{if } t = 0 \end{cases}. \tag{2.12}$$

The parameters τ_+ and τ_- are time constants, while A_+ and A_- are magnitudes that may have some dependence on w_{ij}. The weight update strengthens the synapse if the pre-synaptic spike precedes the post-synaptic event, and vice versa, with an exponentially decaying dependence on the time difference. Unlike the naive Hebb's rule of Equation 2.10, STDP accounts for causal temporal relationships, whereas the former does not consider spike timing. For perfectly coincidental spike arrivals and firings, $t_i^k - t_j^l$ is zero, so there is no weight change with STDP; one cannot be said to have caused the other. For maximal weight change, the pre-synaptic arrival should be slightly before the post-synaptic firing event.

Bengio et al. [4] reinterpret STDP as a machine learning update rule

$$\Delta W_{ij} \propto S_i \dot{V}_j \tag{2.13}$$

where S_i is the pre-synaptic activity and $\dot{V}_j$ is the rate of change of the post-synaptic voltage potential V_j. They note that STDP corresponds to the stochastic gradient descent on the objective function that is improved by the change in post-synaptic potential or ΔV_j.

In machine learning terms, rather than biological ones, this weight update can be stated in terms of the activation function ρ as

$$\delta W_{ij} \propto \rho(s_i)\delta s_j \tag{2.14}$$

where δW_{ij} and δs_j are the differentials of the weight W_{ij} and neuronal activity s_j, respectively.

This particular variant of Hebbian learning, in a contrastive form, is the basis of weight updates in equilibrium propagation.

2.4 Error Functions and Energy-Based Models

Supervised learning in a neural network is driven by the optimization of an objective or error function of the output. A common objective is the minimization of mean squared error (MSE) of the output layer, compared to a target vector output $\vec{y}$

$$\min_{\mathbf{W}} \frac{1}{N} \sum_{j=1}^{N} \left(\hat{y_j} - y_j\right)^2, \tag{2.15}$$

where $\mathbf{W}$ is the set of all weight matrices and

$$\hat{y_j^l} = f_{l,j}\left(\sum_i w_{ij}^l a_i^{l-1}\right), \tag{2.16}$$

l indexes the final and output layer, w_{ij}^l is the connection weight to the j^{th} output neuron from the i^{th} neuron of its preceding layer, $f_{l,j}$ is the activation of this post-synaptic neuron, a_i^{l-1} is the activity of the i^{th} pre-synaptic neuron, y_j is the target of the j^{th} output neuron, and N is the number of output neurons.

Alternatively, minimization of cross-entropy (as in Equation 2.17) between the network output $\hat{y}_i$ and the target y_i is also used, particularly for classification tasks.

$$\min_{\mathbf{w}} -\frac{1}{N}\sum_{i=1}^{N} y_i \ln \hat{y}_i + (1-y_i)\ln(1-\hat{y}_i) \tag{2.17}$$

However, energy-based models are an alternative learning methodology, where the optimization objective is not a function of the output, but rather some scalar "energy" function of the entire network state. An early example of such models is the Hopfield network, shown in Figure 2.2 [19]. The dynamics of the network are governed by the Hopfield energy function of Equation 2.18. The Hopfield energy is

$$E = -\frac{1}{2}\sum_{j,k} w_{jk} a_j a_k - \sum_j x_j a_j \ , \tag{2.18}$$

where a_j and a_k are the activations (states) of nodes j and k, respectively, and x_j is the j^{th} input bit.

While learning, (e.g. binary string) input patterns are memorized by modifying weights such that each pattern is a local minimum of the Hopfield energy function. The storage of such k-bit patterns $\{x_m\}, i = 1, \ldots, k$ occurs during training through weight updates with each x_k as input. Following the negative of the energy function gradient gives the weight update for two connected neurons given M training examples x_m, a_i^m and a_k^m

$$\Delta w_{ij} = \sum_{m=1}^{M}(2a_i^m - 1)(2a_j^m - 1), \qquad w_{ii} = 0, \forall i. \tag{2.19}$$

Consequently, these stored patterns are attractor states of the network. Now, the network may operate with these learned weights, and any (possibly unlearned) input may

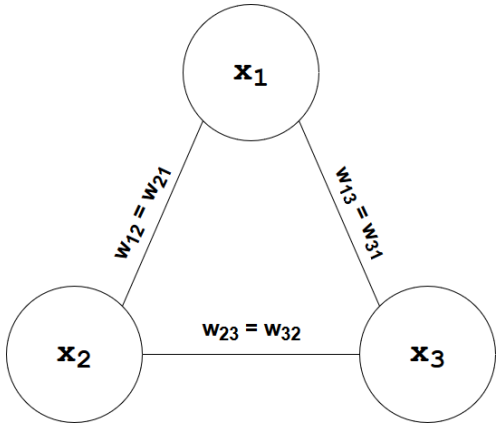


Figure 2.2: A Hopfield network with 3 units. The all-to-all connections are symmetric; i.e. the connection weight w_{ij} from neuron i to neuron j is the same as w_{ji} from neuron j to neuron i.

be applied to the nodes. The states of the nodes will then ideally converge to the learned pattern closest (in Hamming distance) to the input pattern.

Convergence to a (local) minimum is guaranteed, although the "correctness" of this attractor state is not. Given an n-bit input string $\mathbf{x}$, a Hopfield network will converge to a stored $\hat{\mathbf{x}}$ or its additive inverse $-\hat{\mathbf{x}}$ such that the distance between $\mathbf{x}$ and $\hat{\mathbf{x}}$ is minimal. Alternatively, the network may also converge to a *spurious minimum*, which is a non-trained pattern that is a linear combination of an odd number trained patterns.

Energy-based models tend to be more conductive to the use of local learning rules, such as the Hopfield rule in Equation 2.19, due to the fact that weights appear in the total energy function in adjacent pairs corresponding to the pre- and post-synaptic ends of each neuron. Consequently, weight updates performing energy-minimization tend to use neuron-local information only.

Friston proposed the minimization of a quantity he termed "free energy" [9] as the driving force behind neurobiological self-organization for active perception. Active perception refers to the closed-loop sensorimotor setting where sensory model-building and motor control are both involved simultaneously. To quantify sensory predictions, we consider sensory surprise S, or self-information, which is the negative log-likelihood of an event x with probability $P(x)$

$$S(x_i) = -\log(P(x)). \tag{2.20}$$

The the entropy $H(X)$ of a discrete distribution $X = \{x_i\}$ is the weighted sum of these

surprise terms for each of its N events x_i

$$H(X) = -\sum_{i}^{N} P(x_i) \log(P(x_i)). \tag{2.21}$$

The Kullback-Leibler (or KL) divergence is a common measure of the "distance" of distribution Q from distribution P, defined as

$$D_{KL}(P||Q) = -\sum_{i} P(x_i) \log\left(\frac{Q(x_i)}{P(x_i)}\right). \tag{2.22}$$

Free energy may be understood as the sum of either neural activation energies plus entropy, or as statistical surprise plus KL divergence. Free energy is defined as

$$F(x,h) = \underbrace{E_q[-\log P(x,l|\theta)]}_{\text{energy}} - \underbrace{H[Q(\mu|h)]}_{\text{entropy}} = \underbrace{-\log P(x|\theta)}_{\text{surprise}} + \underbrace{D_{KL}[Q(l|h)||P(l|x,\theta)]}_{\text{divergence}} \tag{2.23}$$

where $E[\cdot]$ represents the expected value over the distribution, P is the generative model distribution and Q is the variational density of the external-world latent states μ. The sensory state x can be considered the input in artificial neural network terms, h represents the hidden brain states (equivalent to hidden layers), l represents the latent states of the external world, and θ encapsulates the model parameters (e.g. connection weights in a neural implementation).

The free energy principle works by minimizing F with respect to its parameters, the input sensory state x and the hidden state h. Since D_{KL} is non-negative from Equation 2.22, Friston notes that

$$F(x,h) \geq -\log P(x|h). \tag{2.24}$$

In other words, free energy is an upper bound on surprise.

The sensory model improves predictions, reducing divergence, while the action controller changes the sensory input to reduce prediction error and thus surprise. Thus, energy-based models can help address some of the biological plausibility issues in ANN training, as they provide an alternative to supervised training signals propagating backwards from the output layer into the network.

In addition, according to Friston, such models can account for the fundamentally *bidirectional* processes of sensorimotor or active learning, where both perception and action circuits in the brain are modified. When the environment is viewed as both a sensory input and a modifiable output, as in bidirectional equilibrium propagation, an energy-based view is a more versatile formalism that the supervised input-output model.

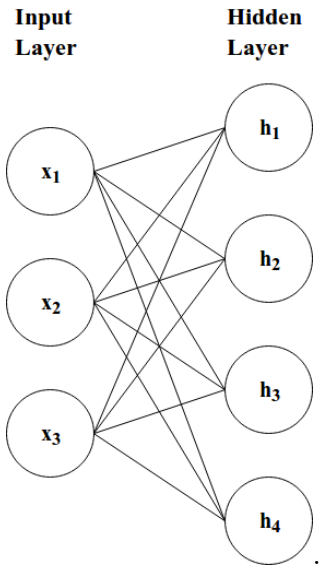


Figure 2.3: A Restricted Boltzmann Machine with 3 input units and 4 output units.

2.5 Contrastive Learning

The origin of the preceding network training methodologies are best understood as credit assignment at the level of neurons. Backpropagation uses the chain rule to determine the steepest descent path for a connection weight, while Hebbian learning arises from local correlations. On the other hand, contrastive methods are based on the idea of matching two population distributions. A network is run in two phases, a *free* phase without external influence and a *clamped* or externally-forced phase, and the distributions of neuronal activities of these two phases are used to determine the gradient for weight updates.

The initial success of this approach came primarily from contrastive divergence (CD) applied to Restricted Boltzmann Machines (RBMs) [24]. Essentially, RBMs are a variant of Hopfield networks with hidden units connected as a bipartite graph. Consequently, RBMs have a Hopfield-like energy function, as defined in Equation 2.25 in terms of visible ("input") and hidden neurons x and h, respectively, as well as their respective weights w_{ij} and biases a_i and b_j.

$$E(\mathbf{x}, \mathbf{h}, \mathbf{W}, \mathbf{a}, \mathbf{b}) = -\sum_i a_i x_i - \sum_j b_j h_j - \sum_i \sum_j w_{ij} x_i h_j \tag{2.25}$$

Ideally, optimizing an RBM would mean minimizing the gradient of a loss function

defined as the difference between the equilibrium state energies of two different phases of the network

$$\frac{\partial L(\theta; x)}{\partial \theta} = \underbrace{\left\langle \frac{\partial E(\mathbf{x}, \theta)}{\partial \theta} \right\rangle_{\infty}}_{\text{positive phase}} - \underbrace{\left\langle \frac{\partial E(\mathbf{x}, \theta)}{\partial \theta} \right\rangle_{0}}_{\text{negative phase}} \tag{2.26}$$

where θ encapsulates the parameters of the network (i.e. any weights and biases). For RBMs, $\langle \cdot \rangle_0$ represents an average with respect to the sample (i.e. dataset) distribution while $\langle \cdot \rangle_\infty$ represents an average with respect to model (i.e. network-generated) distribution. By optimizing the loss function L, the two equilibrium distributions are made more similar. However, this is difficult to compute, with a large variance in the estimate for the positive phase component [5].

Instead, RBMs are trained using contrastive divergence to minimize the distance between two Kullback-Leibler divergences [5]

$$CD_n = D_{KL}(P_0 || P_\infty) - D_{KL}(P_n || P_\infty), \tag{2.27}$$

where n specifies the number of iterations of the contrastive divergence procedure. Specifically, P_0 is the data distribution and P_∞ is the model distribution parametrized by the weights $\mathbf{W}$ so that

$$D_{KL}(P_0 || P_\infty) = \sum_{\mathbf{x}} P_0(\mathbf{x}) \log \left(\frac{P_0(\mathbf{x})}{P(\mathbf{x}, \mathbf{W})} \right). \tag{2.28}$$

From Equation 2.25, gradient ascent on this objective then leads to the update

$$\frac{\partial \log P(\mathbf{x})}{\partial w_{ij}} = \langle x_i h_j \rangle_0 - \langle x_i h_j \rangle_\infty \tag{2.29}$$

$$P(\mathbf{x}) = \frac{\sum_{\mathbf{h}} e^{-E(\mathbf{x}, \mathbf{h})}}{\sum_{\mathbf{v}, \mathbf{h}} e^{-E(\mathbf{x}, \mathbf{h})}}. \tag{2.30}$$

In practice, it is not P_∞ that is computed but rather an approximation P_n using n iterations. A single iteration of contrastive divergence (CD-1) for a data point $\mathbf{x}$ performs the following steps:

1. Compute the hidden unit activations $\mathbf{h}$ for input $\mathbf{x}$ given weights $\mathbf{W}$ and hidden layer biases $\mathbf{b}$
2. Compute and store the positive phase gradient $\mathbf{x}\mathbf{h}^T$

3. Generate a reconstructed input $\hat{\mathbf{x}}$ using $\mathbf{h}$, $\mathbf{W}$ and visible layer biases $\mathbf{a}$, then use the reconstructed input to compute new hidden layer activities $\hat{\mathbf{h}}$

4. Compute and store the negative phase gradient $\hat{\mathbf{x}}\hat{\mathbf{h}}^T$

5. With some learning rate ϵ, update the weights $\mathbf{W}$ and biases $\mathbf{a}$ and $\mathbf{b}$ (for the visible and hidden layers, respectively) according to

$$\Delta\mathbf{W} = \epsilon(\mathbf{x}\mathbf{h}^T - \hat{\mathbf{x}}\hat{\mathbf{h}}^T) \tag{2.31}$$

$$\Delta\mathbf{a} = \epsilon(\mathbf{x} - \hat{\mathbf{x}}) \tag{2.32}$$

$$\Delta\mathbf{b} = \epsilon(\mathbf{h}^T - \hat{\mathbf{h}}^T) \tag{2.33}$$

Sutskever and Tieleman [42] show that contrastive divergence does not follow the gradient of any objective function. While this does not preclude it from minimizing a function without following a gradient, the authors do show that convergence can be prevented by adding certain regularization terms. In practice, this can make training via contrastive divergence a matter of careful parameter selection.

Nevertheless, Xie and Seung [45] show that the training and test error due to contrastive Hebbian learning and backpropagation are similar for a layered network with forward as well as backward connections. Contrastive Hebbian learning is a similar local learning rule with a Hebbian (i.e. weight-strengthening) update followed by an anti-Hebbian (i.e. weight-weakening) update.

Hinton [17] notes that contrastive learning forces weight symmetry as long as there is weight decay proportional to weight magnitude. As the weight updates due to contrastive divergence are symmetric (i.e. $\Delta w_{ij} = \Delta w_{ji}$, the weights w_{ij} and w_{ji} always approach each other.

Equilibrium propagation uses a variant of a contrastive Hebbian algorithm to sample the network at different levels of external forcing or clamping. In such a layered network, contrastive methods allow errors to be propagated from terminal layers, which have access to environmental inputs, to internal layers.

2.6 Equilibrium Propagation

A recent learning algorithm for the supervised training of continuous-time, energy-based neural networks is equilibrium propagation [37]. It is a phased algorithm; the *free* phase

presents the input to the network and allows it to settle into a *free equilibrium*, and the second, *weakly-clamped* phase then introduces an error to the output neurons (whose magnitude depends on the degree of clamping) and the network settles into a *weakly clamped equilibrium*. The weakly-clamped phase is essentially a small perturbation, forcing the network away from the free equilibrium in the direction that improves task performance. A contrastive Hebbian-like local learning rule then uses the difference in co-activation statistics of neurons between the two phases to make a weight update.

Energy Function & Activity Dynamics

Equilibrium propagation is built around a continuous-time neural model. Consequently, it is defined by differential equations. The activity s_i of each neuron is governed by a network energy function

$$F(\mathbf{w}, \mathbf{x}, \mathbf{y}, \beta, \mathbf{s}) = E(\mathbf{w}, \mathbf{x}, \mathbf{s}) + \beta C(\mathbf{w}, \mathbf{x}, \mathbf{y}, \mathbf{s}) \tag{2.34}$$

where $\mathbf{w}$ is the connection weights, $\mathbf{x}$ is the input to the network, $\mathbf{y}$ is the target output, β is the clamping factor, and $\mathbf{s} = \{s\} = \{x, h, \hat{y}\}$ is the collection of neuron states, comprised of input, hidden and output neurons, respectively. The Hopfield-like total energy function is composed of two sub-parts: the internal energy E and the external energy or "cost" C, modulated by the clamping factor $\beta > 0$.

Each neuron with activity s_i attempts to minimize the total energy. This process is completely local. Neural activity follows the negative of the energy gradient,

$$\frac{ds_i}{dt} = -\frac{\partial F}{\partial s_i} = -\frac{\partial}{\partial s_i}(E + \beta C) = -\frac{\partial E}{\partial s_i} - \beta \frac{\partial C}{\partial s_i}. \tag{2.35}$$

A simple substitution of the neuronal dynamics shows that following Equation 2.35 guarantees that the total energy F is non-increasing in time

$$\frac{dF}{dt} = \frac{\partial F}{\partial \mathbf{s}} \cdot \frac{d\mathbf{s}}{dt} = -\left|\left|\frac{d\mathbf{s}}{dt}\right|\right|^2 \leq 0. \tag{2.36}$$

Neurons implement the hard sigmoid activation function

$$\rho(s_i) = \min(1, \max(0, s_i)). \tag{2.37}$$

Biologically, the state or activity s_i of a neuron would correspond to its average membrane potential, while $\rho(s_i)$ would be the firing rate.

The total internal energy on the network is

$$E(s) = \frac{1}{2}\sum_i s_i^2 - \sum_{i\neq j} W_{\mathrm{FW},ij}\rho(s_i)\rho(s_j) - \sum_{i\neq j} W_{\mathrm{BW},ij}\rho(s_i)\rho(s_j) - \sum_i b_i\rho(s_i) \tag{2.38}$$

where $W_{FW,ij}$ is the connection weight matrix for all upper-layer neurons s_j to which s_i outputs ("projects"), $W_{BW,ij}$ is the same for all lower-area projections, and b_i is the bias of s_i. Equation 2.38 shows that E can be split into a quadratic leaky (i.e. decay) term and the weighted sum of input currents to the neuron.

The external energy on the network is task-dependent. In the case of the classification problems presented in Scellier et al. [37] as well as this thesis, it is defined as the L^2 norm

$$C(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{2}||\hat{\mathbf{y}} - \mathbf{y}||^2 \tag{2.39}$$

where $\hat{\mathbf{y}}$ is the vector activities of output neurons $\{\hat{y}_i\} \in \mathbf{s}$, and $\mathbf{y}$ is the one-hot encoded vector of the output class y (i.e. with only one nonzero entry corresponding to the class).

The forces applied to each neuron's activity s_i due to the potential energies E and C are given by their derivatives. The force due to the internal energy is

$$\frac{\partial E}{\partial s_i} = \underbrace{s_i}_{\text{leaky term}} - \underbrace{\rho'(s_i)}_{\text{slope of activation function}} \left(\underbrace{\sum_{j\neq i} W_{ij}\rho(s_j) + b_i}_{\text{input current}} \right). \tag{2.40}$$

As summarized in Equation 2.40, the internal force can be broken down into a term due to persistent activity s_i (i.e. the pre-existing neuronal state), and a term due to incoming input to the neuron; these neurons behave as leaky integrators. Likewise, the "external force" due to the output cost function is present only when β is non-zero, and affects only output neurons y_i, since

$$\frac{\partial C}{\partial s_i} = \begin{cases} 0 & \text{if } s_i \notin \{\hat{y}\} \\ s_i - y_i & \text{if } s_i \in \{\hat{y}\} \end{cases} \tag{2.41}$$

Activation Function & Dead Neurons

The derivative of the hard sigmoid activation function is defined such that it keeps neurons operating in the linear region $0 \leq s_i < 1$. If the activity drops below zero, the contribution

of the input current to the internal force is eliminated, allowing the leaky term in Equation 2.40 to raise the activity up to zero. This serves to minimize the *dead neuron problem*, where a zero activation function derivative prevents useful updates to the neuron state, effectively making the neuron a uniformly zero-output computational burden. For large values of s_i, the slope of the activation function is zero, so the gradient $\frac{\partial E}{\partial s_i} = s_i$ and, following the negative of this gradient (Equation 2.36, s_i decreases over time. This prevents the complimentary "exploding gradient problem", where the neuronal activity increases without bound. To minimize this issues, the derivative of ρ is defined at the boundaries as

$$\rho'(s_i) = \begin{cases} 0 & s < 0 \\ 1 & 0 \leq s \leq 1 \\ 0 & s \geq 1 \end{cases}. \tag{2.42}$$

EP Algorithm

Equilibrium propagation can be summarized as operating over three timescales, and it is important to identify what parameters are varied at what scale.

1. The fast neuronal dynamics descend along the gradient of the total energy $\frac{\partial \mathbf{F}}{\partial s_i}$.
2. The slower network dynamics run until the network converges to an equilibrium state $\{s^0\}$. This is repeated for a free network followed by a non-zero clamping factor β, giving the weakly-clamped equilibrium state $\{s^\beta\}$.
3. At the end of each equilibrium propagation cycle, the contrastive weight update

$$\Delta W_{ij} = \frac{\lambda}{\beta}\left(\left(s_i^\beta\right)\rho\left(s_j^\beta\right) - \left(s_i^0\right)\rho\left(s_j^0\right)\right) \tag{2.43}$$

   is applied, where λ is the learning rate.

2.7 Timescale and Gradient Dynamics

Given the complexity of biological nervous systems, particularly the brain, many phenomena have nuanced explanations covering multiple timescales. Inter-neuron competition, for example, can be framed in terms of synaptic learning and sparsity, or alternatively in terms

of activity and input selectivity [32]. The temporal and spatial level of a neural model is an important consideration for neurobiological modeling.

Bassett & Sporns [3] make a distinction, in terms of network science, between dynamics *on* networks (i.e. the changing patterns of node activity on a graph), and dynamics *of* networks (i.e. changes in edges, and consequently in the structure of the network). In neuroscience and neural networks, these correspond to neural activity and synaptic plasticity, respectively. Of course, both phenomena occur simultaneously, with synaptic plasticity occurring over longer timescales. In biological neurons, this would be an online process, while, in artificial neural networks, batch processing may occur.

In the context of biological plausibility, the relevant consideration is the type of computation neurons perform at each stage. In backpropagation, forward and backward passes correspond to activity and synaptic plasticity, yet they require the same neurons and connections to function differently. In Hebbian and energy-based models, including equilibrium propagation, neurons perform the same computations in each stage, and weight changes do not require propagation through the network.

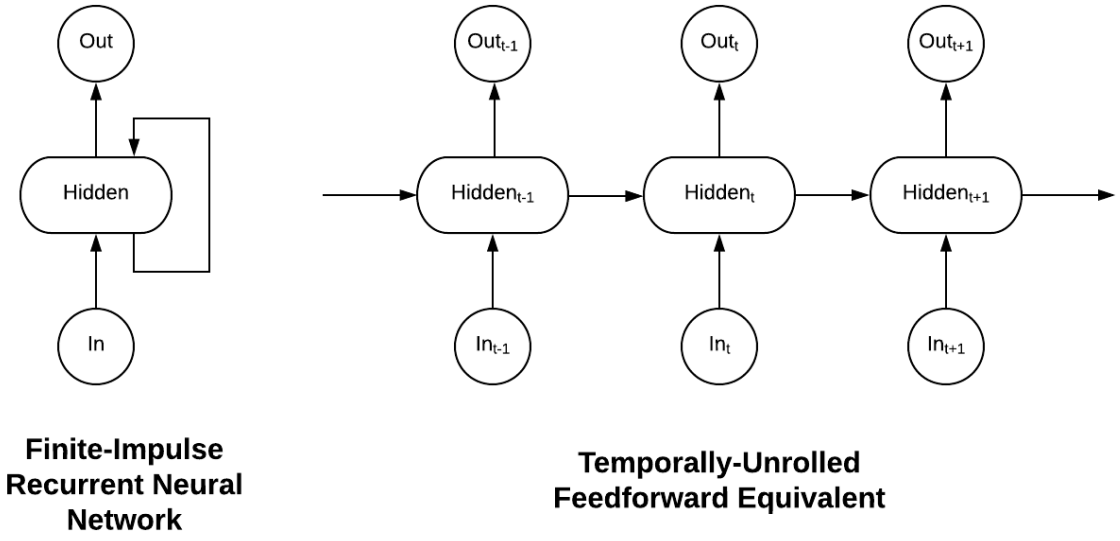
More comprehensive neural models would likely also include other biological processes with functional, computational effects. For example, at longer time scales intrinsic plasticity, the homeostatic modification of a neuron's activation function to maximize information transfer [43], would show its effects. At higher levels still, gene expression, organism development, and even evolution influence the behaviour of neural systems.

Since training a neural network has the goal of optimizing task performance, weight updates typically follow the gradient of an objective function (or, equivalently, the negative of the gradient of an error). This is true by definition in the case of algorithms such as backpropagation, as evident in Equation 2.1.

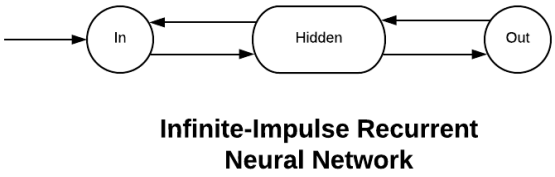
2.8 Generative Models and Recurrence

Recurrent neural networks (RNNs) may be classified by the duration of their impulse response. Figure 2.4a depicts an example of a finite-impulse RNN. The output of a finite-impulse system to a finite impulse (i.e. an input applied for a short duration) should decay to zero. Notably, an finite-impulse RNN's cycle is directed; consequently it can be represented by an equivalent feedforward network. However, infinite-impulse networks may sustain activities that never quite decay to zero. This happens specifically in undirected graphs such as Figure 2.4b, as there is no activity sink.

Biologically, recurrent, lateral connections and feedback are extremely common. In particular, tasks such as attention and input selectivity seem to require top-down feedback. Even feedforward models of biological systems acknowledge the presence of feedback from higher areas in the brain [46]. As such, we can conclude that biologically relevant network architectures should not only be recurrent, but that they should also utilize this recurrence for some sort of task.



(a) Directed cyclic RNNs may be unrolled.



(b) This network forms an undirected cylic graph and cannot be unrolled into a feedforward network.

Figure 2.4: *Impulse Response and Directedness*

Chapter 3

Bidirectional Equilibrium Propagation

Bidirectional equilibrium propagation (BEP), the main contribution of this thesis, is a variant of equilibrium propagation that works by alternating between forward and backward passes. Unlike backpropagation-like algorithms, however, both directions use the same neuronal computations, and weight updates are always computed as the contrastive co-activations of two runs of the network with different levels of clamping. Simultaneously, bidirectional equilibrium propagation allows recurrent connections. Unlike the original equilibrium propagation model, bidirectional equilibrium propagation uses a directed graph model with distinct (asymmetric) forward and backward weights.

3.1 Network Architecture

For clarity of nomenclature, each sequence of free and fixed phase passes, culminating in a weight update, is called one *training pass*. Each time the network is allowed to settle to an equilibrium, with a new input or a different clamping factor is referred to as a *phase*, as with regular equilibrium propagation. For the tasks used in this section, the discriminative, "forward", "bottom-up", or "afferent" direction classifies a $D-$dimensional vector input "image" into one of D classes. The generative, "backward", "top-down", or "efferent" direction generates an image given a class as input. Both directions use the same network, with bidirectional connections between layers; the difference is what layer is clamped. In the discriminative mode, the bottom or $x-$layer is clamped, while the top or $y-$layer is clamped for generative operation.

In bidirectional equilibrium propagation, there are task-specific input and output units that alternate roles, as well as hidden units. The BEP architecture is distinct from standard layered neural networks, Hopfield networks with all-to-all connections, as well as restricted Boltzmann machines. Thus, to avoid confusion, especially for bidirectional equilibrium propagation networks, neither terminal layer of a network is considered a purely "input" or "output" layer without a directional qualifier. As shown in Figure 3.1, discriminative mode uses the bottom layer for inputs and the top layer for output, while this is reversed for the generative mode. .

In addition, it is notable that BEP networks are undirected graphs. This makes them unsuitable for recurrent variants of backpropagation, as temporally expanding the network does not remove cycles.

3.1.1 Energy Dynamics

Learning in the top-down direction requires augmenting the energy function of Equations 2.38 and 2.39 with a generative cost function D defined as

$$D(\hat{\mathbf{z}}, \mathbf{z}) = \frac{1}{2}||\hat{\mathbf{z}} - \mathbf{z}||^2 \tag{3.1}$$

where $\hat{\mathbf{z}}$ is the output layer activities of the generative mode and $\mathbf{z}$ is the target class archetype (described in Section 3.3). Notably, while the squared L^2 norm was applied to a one-hot encoded target, the interpretation for this cost D is the distance from the class archetype vector defined in Section 3.3 at the equilibrium (generative) output state of the network. The gradient of this new external energy is nevertheless the same as Equation 2.41

$$\frac{\partial D}{\partial s_i} = \begin{cases} 0 & \text{if } s_i \notin \{\hat{z}\} \\ s_i - z_i & \text{if } s_i \in \{\hat{z}\} \end{cases} \tag{3.2}$$

with $\{\hat{z}\}$ being the entirety of the set of neurons in the bottom layer.

Combining this into the total energy gives

$$F(\theta, \mathbf{z}, \mathbf{y}, \alpha, \beta) = E(\theta, \mathbf{s}_\theta^{\alpha,\beta}) + \beta C(\mathbf{s}_\theta^{\alpha,\beta}, \mathbf{y}) + \alpha D(\mathbf{s}_\theta^{\alpha,\beta}, \mathbf{z}) \tag{3.3}$$

where α is the clamping factor of this generative cost function, θ encapsulates the network parameters (weights and biases), $\mathbf{z}$ is the bottom layer external target and $\mathbf{y}$ is the top layer external target.

Considering the discriminative and generative cases, we define the free-phase equilibria, which is effectively the "prediction" of the model parametrized by θ with input but not target clamping, as $\mathbf{s}_\theta^{\alpha=0,\beta=\infty}$.

The total energy of the network is

$$F = \frac{1}{2}\sum_i s_i^2 - \sum_{i\neq j} w_{\text{FW}.ij}\rho(s_i)\rho(s_j) - \sum_{i\neq j} w_{\text{BW}.ij}\rho(s_i)\rho(s_j) - \sum_i b_i\rho(s_i) + \frac{\beta}{2}||\hat{\mathbf{y}}-\mathbf{y}||^2 + \frac{\alpha}{2}||\hat{\mathbf{z}}-\mathbf{z}||^2. \tag{3.4}$$

where the separate forward and backward weights are defined as for Section 2.6.

The gradient of the total energy, and consequently the force on each neuron s_i, is thus

$$\frac{\partial F}{\partial s_i} = s_i - \rho'(s_i)\left(\sum_{j\neq i} W_{ij}\rho(s_j) + b_i\right) + \beta\frac{\partial C}{\partial s_i} + \alpha\frac{\partial D}{\partial s_i}. \tag{3.5}$$

When a layer is fully clamped, as is the case for the bottom layer which is clamped to $\mathbf{x}$ during the regular discriminative mode of equilibrium propagation, the clamping factor α is essentially infinite. In this case, the rest of the network can have no influence on the input layer, and by convention the D term drops out of the total energy gradient, which reduces to the F in Section 2.6. This is illustrated in Figure 3.1.

Gradient Dynamics

Equilibrium propagation allows the network to settle into a free equilibrium, then clamps the output and allows the network to settle into a (weakly-)clamped or perturbed equilibrium. The direction of weight updates are intended to shift the free equilibrium point (in the space of network activities $\{s_i\}$) to the clamped equilibrium point. Basically, after the weight update, the free dynamics of the network should settle to a point that minimizes output error. This directional information would not be possible without running the network in two phases per task.

3.2 Discretized Simulation

As equilibrium propagation uses a continuous-time neural model, in each phase the dynamical system with a state defined by the set of activities of all neurons $\mathbf{s}$ is simulated.

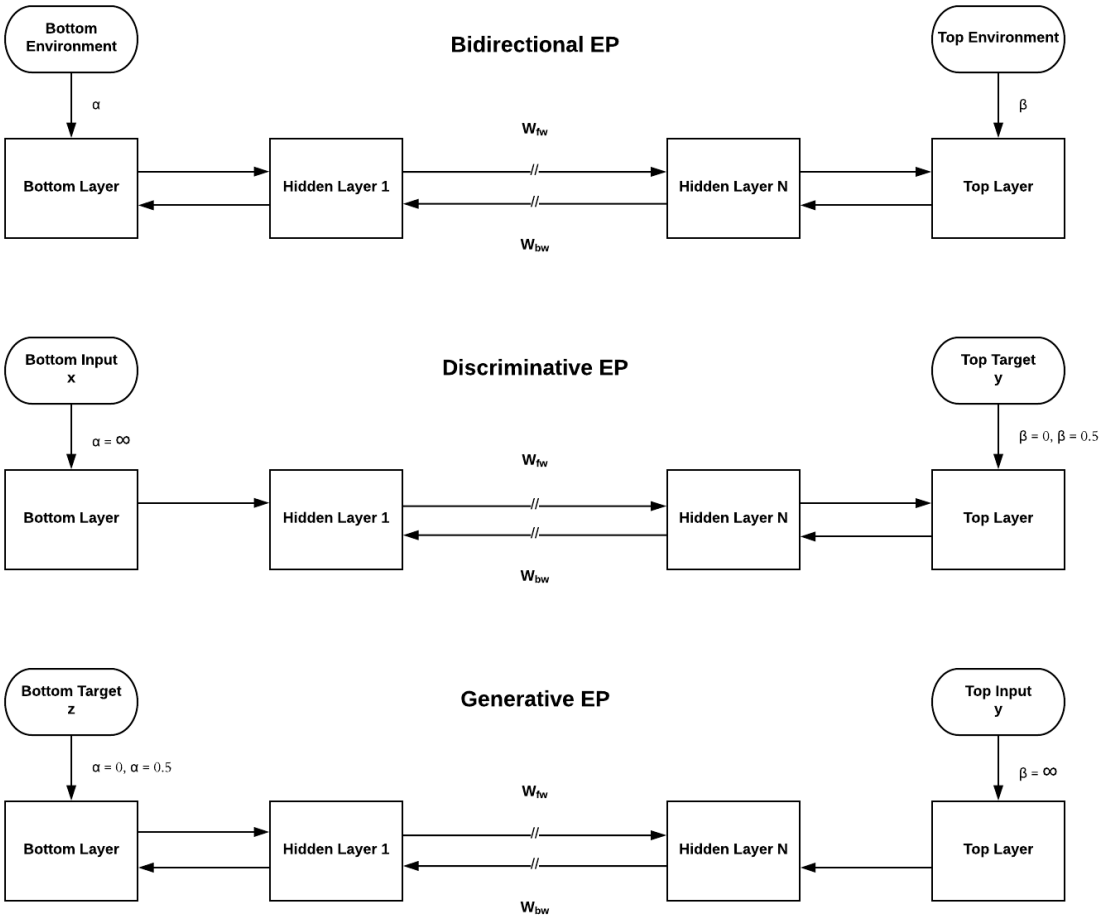


Figure 3.1: *Bidirectional Equilibrium Propagation:* The clamping factors α and β determine the effect of the environment on the bottom and top input/output layers, respectively. When an input (on either side: top or bottom) is fixed, the clamping factor is infinite so that layer is not affected by network dynamics.

Using Euler's method to solve the differential equation in Equation 2.35, with a time step ϵ

$$s_i(t+\epsilon) = s_i(t) - \epsilon \frac{\partial F}{\partial s_i}(t). \tag{3.6}$$

Neuron activities and biases are initialized to zero, while weights are initialized using a Xavier/Glorot uniform distribution, suggested by Glorot and Bengio [15].

```
def initialize_weights(...):
  n_lower = size(layer[l])
  n_higher = size(layer[l+1])
  for each lower_neuron in n_lower:
    for each higher_neuron in n_higher:
      // Uniform Glorot initialization
      max_value = sqrt(6 / (n_lower + n_higher))
      W_fw[in] = unif( - max_value, + max_value )
      W_bw[in] = unif( - max_value, + max_value )
  end
```

Learning rates are assigned individually by weight layer. Heuristically, learning performs better when learning rates are increased further from the source of external influence, to address the vanishing gradient problem. In a purely discriminative task, this means that earlier levels will have higher learning rates than later areas, and vice versa for a purely generative task.

3.3 Dataset

The network was tested on an artificial dataset containing $D-$dimensional vectors and their corresponding classes, using the following process.

1. A large number of $D-$dimensional vectors are sampled with components in the range $[0, 1)$. These vectors are clustered into D clusters, with the centroids forming the class archetypes.

2. A cluster centroid is randomly chosen, and zero-mean Gaussian noise with standard deviation of 0.1 is added to give a data point $\mathbf{x}_i$.

3. Each data point $\mathbf{x}_i$, along with its corresponding cluster label y_i and class archetype $\mathbf{z}_i$ form one training example $(\mathbf{x}_i, y_i, \mathbf{z}_i)$.

The pair $(\mathbf{x_i}, y_i)$ is the training data for the discriminative direction, while $(y_i, \mathbf{z_i})$ is the data point for the generative direction.

3.3.1 Learning Algorithm

To derive the weight update function, we can consider a contrastive learning objective function. Such a learning objective is interpreted as bringing two equilibrium distributions closer together. Specifically, these are the output layers of the network at equilibrium, with and without target clamping. For unidirectional EP, we can define this objective function as

$$J_{EP} = \beta(C_\infty^{\beta_+} - C_\infty^{\beta_-}) \tag{3.7}$$

where C_∞ is the external cost or energy of the equilibrium states with the respective clamping factor values (α_+, β_+) and (α_-, β_-).

Extending this to BEP requires incorporating the generative cost, so we define a new vector objective function

$$\vec{J}_{BEP} = \begin{bmatrix} \beta(C_\infty^{\alpha_+\beta_+} - C_\infty^{\alpha_-\beta_-}) \\ \alpha(D_\infty^{\alpha_+\beta_+} - D_\infty^{\alpha_-\beta_-}) \end{bmatrix} \tag{3.8}$$

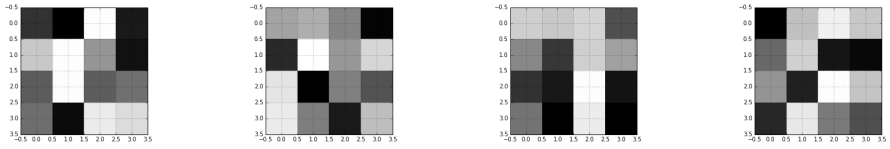
$$\frac{\partial J}{\partial \theta} = \frac{\partial F^{\alpha_+\beta_+}}{\partial \theta} - \frac{\partial F^{\alpha_-\beta_-}}{\partial \theta} \tag{3.9}$$

$$\frac{\partial J}{\partial \theta} = \frac{\partial F^{\alpha_+\beta_+}}{\partial \theta} - \frac{\partial F^{\alpha_-\beta_-}}{\partial \theta} \tag{3.10}$$

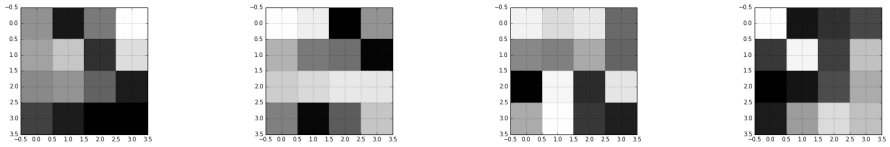
To perform a contrastive weight update on the network, two (or more) equilibrium states $\mathbf{s}$ with different values of the clamping factors (α, β) must be sampled. Firstly, the edge case of a completely unclamped network with $\alpha = 0, \beta = 0$ is meaningless, as there is no input to the network. Similarly, both the inputs and outputs cannot simultaneously be fully clamped; the parameters $\alpha = \infty, \beta = \infty$ allow neither discriminative nor generative learning.

For simplicity, we use the 4 points with clamping factors $(0, \infty)$, (α, ∞), $(\infty, 0)$ and (∞, β).

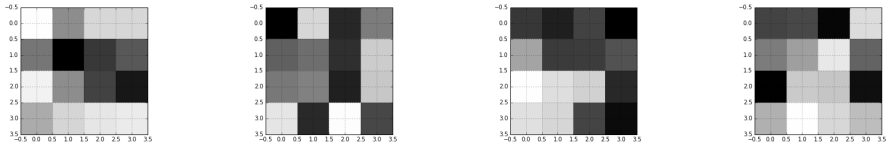
$$\Delta W_{ij} = \frac{\lambda}{\alpha}\left(\left(s_i^\alpha\right)\rho\left(s_j^\alpha\right) - \left(s_i^0\right)\rho\left(s_j^0\right)\right)_{\beta=\infty} + \frac{\lambda}{\beta}\left(\left(s_i^\beta\right)\rho\left(s_j^\beta\right) - \left(s_i^0\right)\rho\left(s_j^0\right)\right)_{\alpha=\infty} \tag{3.11}$$



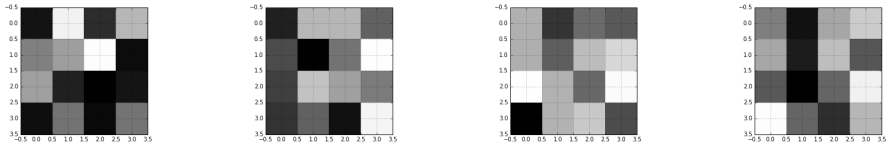
(a) Class 1 (b) Class 2 (c) Class 3 (d) Class 4



(e) Class 5 (f) Class 6 (g) Class 7 (h) Class 8



(i) Class 9 (j) Class 10 (k) Class 11 (l) Class 12



(m) Class 13 (n) Class 14 (o) Class 15 (p) Class 16

Figure 3.2: Example class archetypes for $D = 16$. Data $\mathbf{x}$ vectors are generated by adding independent and identically-distributed Gaussian noise to a class archetype.

Chapter 4

Results and Discussion

4.1 Results

Bidirectional equilibrium propagation successfully and simultaneously learns the discriminative and generative transformations for the artificial dataset, by achieving 0% validation error on the generative task, and low (below 1%) validation error on the discriminative task. In addition, special cases of the network were also tested to isolate the mechanisms contributing to this bidirectional learning, namely

- asymmetric vs. symmetric (i.e. independent vs. matching) forward and backward weights,
- discriminative-only training vs. generative-only training compared to bidirectional learning, and
- BEP vs. restricted BEP (r-BEP), where only the weights corresponding to the task direction are modified (note that this is distinct from regular EP).

For each task, which includes several permutations of the above scenarios, the network is run in the free phase (i.e. with no target-clamping) until equilibrium. The equilibrium states of the terminal layer are then read out and compared to the target. The discriminative classification task presents the input $\mathbf{x}$ at the bottom layer the network and evaluates the top-layer output in comparison to the target y. The generative task presents the class y at the top layer and evaluates the similarity of the generated bottom-layer vector to the target $\mathbf{z}$.

The works discussed in Section 2.2 on feedback alignment and random backpropagation weights suggest that optimizing even a restricted subset of weights can improve task performance. Carrying this idea to bidirectional learning with asymmetric weights, optimizing forward and backward weights for the discriminative-only error improves the generative task performance significantly. Optimizing the generative task does not improve the discriminative task as much, although some gains are still evident. However, this transfer learning is lost entirely when symmetric weights are used.

For all cases, discriminative and generative modes were both run for 25 free phase iterations and 5 clamped iterations each, with $\alpha = 0.5, \beta = 0.5$ and the timestep $\epsilon = 0.5$. The layer-wise learning rates were set at $lr = (0.1, 0.05)$, and the network was trained for 20 epochs.

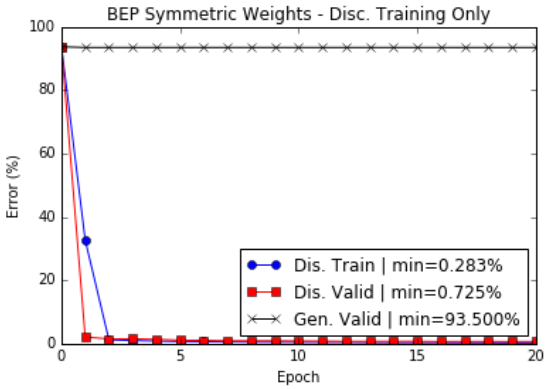
4.1.1 Bidirectional Equilibrium Propagation

Figure 4.1 shows the classification errors for the special case of "tied", or symmetric, forward and backward weights. In addition to bidirectional training in Figure 4.1c, two other cases are shown, of only discriminative learning in Figure 4.1a and only generative learning in Figure 4.1b. This shows that transfer learning across the tasks is normally low; training for near-perfect discriminative performance does not improve generative performance to better than chance (i.e. 1/16 success rate or 93.75% error), and likewise for generative training. Yet, bidirectional training improves both objectives without compromise.

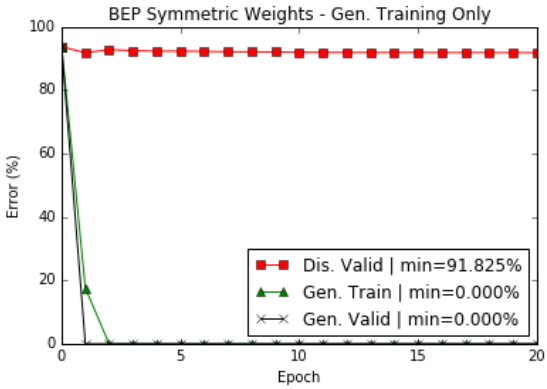
Likewise, the split or asymmetric weights case in Figure 4.2 optimizes both tasks well. Additionally, the split weights afford some freedom for the complimentary task to improve even though it is not being optimized. Generative performance improves significantly when only discriminative performance is optimized, and discriminative performance improves slightly better than chance with only generative training. This asymmetry in transferability may have some dependence on the separability of the dataset, but it indicates that the two tasks may be encouraging different hidden-layer representations, with varying applicability to the complimentary task.

4.1.2 Restricted Bidirectional Equilibrium Propagation

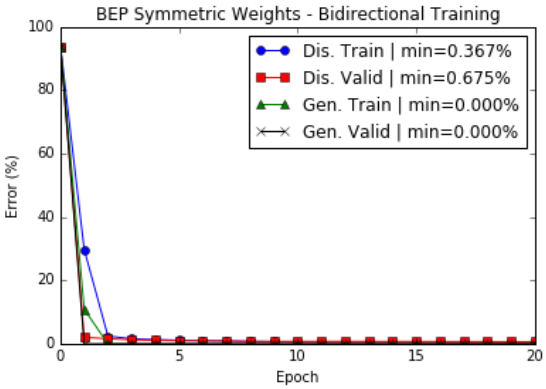
To further isolate the mechanisms in BEP, the special case of r-BEP with only direction-specific learning for each task is shown in Figure 4.3. For the discriminative-only case in Figure 4.3a, only bottom-up weights are adjusted to optimize the discriminative task. Nevertheless, generative performance improves substantially. Figure 4.3b shows the case



(a) Discriminative with tied weights

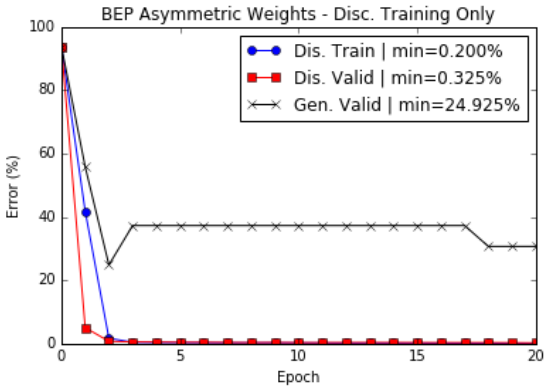


(b) Generative with tied weights

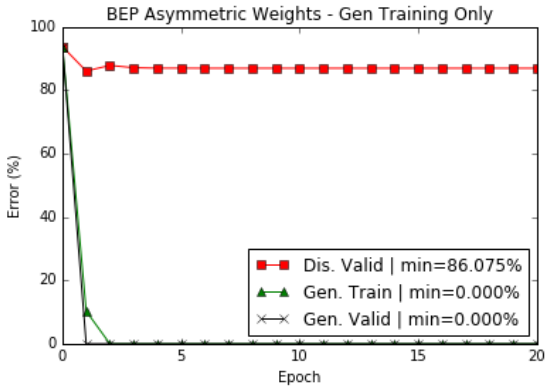


(c) Bidirectional with tied weights

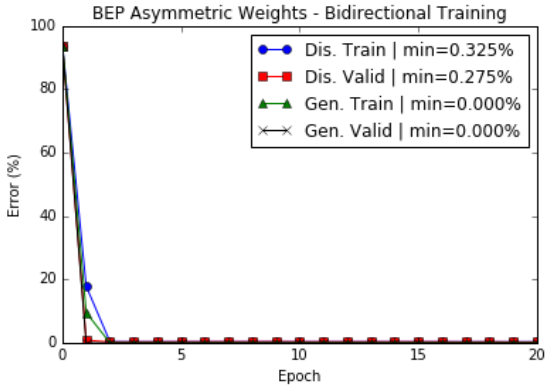
Figure 4.1: Learning in BEP network with tied weights.



(a) Discriminative with split weights



(b) Generative with split weights



(c) Bidirectional with split weights

Figure 4.2: Learning in BEP network with asymmetric (split) forward and backward weights.

of generative learning to adjust only top-down weights. Once again, discriminative performance does not improve much, although it does better than chance.

Even though top-down weights are not explicitly optimized in the discriminative-only case of Figure 4.3a, it must be noted that these weights still influence the network activity dynamics through the network energy function from Equation 3.3.

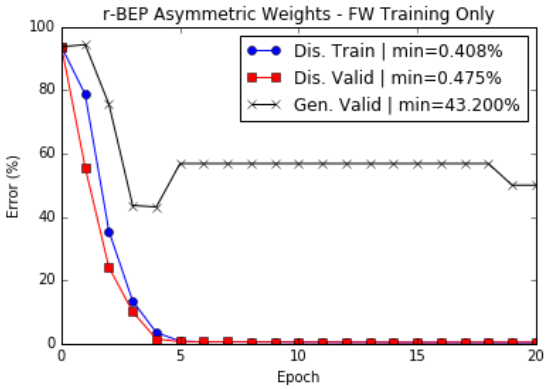
4.2 Biological Plausibility

Equilibrium propagation is consistent with O'Reilly's criteria in Section 2.1. The model is parametrized by a distributed connection weights that are modified by local Hebbian plasticity, while avoiding biologically implausible constraints such as linear error propagation. Simultaneously, equilibrium propagation is able to perform error-driven supervised learning, as proven by the original implementation on the MNIST digit classification dataset [37].

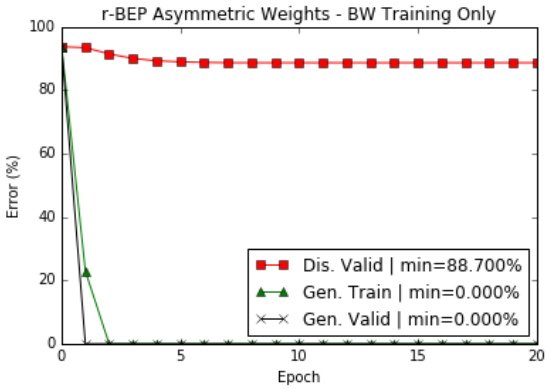
Most "biological" neural learning rules, such as Hebbian learning and STDP, are unsupervised. Brains constantly deal with a flood of inputs with no associated labels, categories or target outputs, and efforts to understand synaptic plasticity cannot ignore the high degree of self-organization evident in biological neural networks [2]. A great deal of synaptic plasticity and neuronal specialization can be explained by Hebbian learning without the use of supervised error signals [18]. In fact, the biological implausibility of algorithms like backpropagation comes specifically and explicitly from the credit assignment processing due to an external error.

In addition, bidirectional equilibrium propagation extends this error-driven learning to simultaneously learn discriminative and generative models for a supervised dataset. This more explicitly aims to replicate the sort of feedback and lateral processing that occurs in the brain. Typically, neurons are often described as "encoding" a certain property of the input. BEP takes some first steps towards a more sophisticated idea of neural input-selectivity than the classical idea of receptive fields.

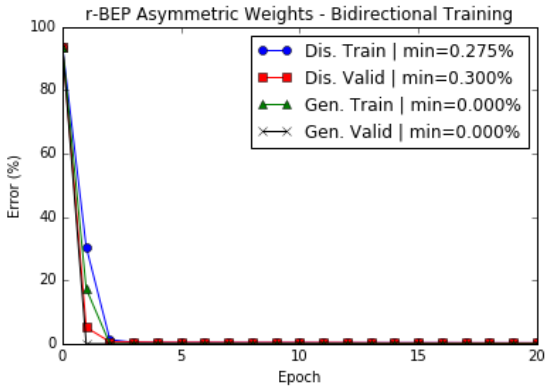
BEP also removes the biologically implausible constraint of symmetric forward and backward weights. It extends the layer-wise level of recurrence to a network-level task-reversibility, by attempting to learn the corresponding generative task for a discriminative problem. The same set of parameters are shared across forward and backward tasks; this transfer learning and multiplicity of encoding is another layer of distributed representation across (related) tasks.



(a) Discriminative with split weights



(b) Generative with split weights



(c) Bidirectional with split weights

Figure 4.3: Learning with r-BEP and split forward and backward weights.

The phased contrastive update does not necessarily require the storage of free and weakly-clamped equilibrium states in a biological implementation. As framed by Xie and Seung [45], contrastive update of Equation 2.43 can be viewed as two separate weight updates

1.
$$\Delta W_{ij-} = -\frac{\lambda}{\beta} s_i^0 \rho\left(s_j^0\right) \tag{4.1}$$

2.
$$\Delta W_{ij+} = \frac{\lambda}{\beta} s_i^\beta \rho\left(s_j^\beta\right) \tag{4.2}$$

which would be local signals, both spatially and temporally.

Additionally, Baldi and Pineda [1] have proposed neurobiological mechanisms for the phase-switching of contrastive learning. Slower timescale neural oscillations can possibly account for the modification and synchronization of environmental clamping factors α and β.

Chapter 5

Conclusion

Equilibrium propagation proposes a local mechanism for neural dynamics based on energy-minimization followed by a local, contrastive Hebbian weight update. However, the original equilibrium propagation model was restricted to only feedforward operation with symmetric feedback weights. This thesis generalizes equilibrium propagation to multiple output layers and asymmetric feedforward-feedback weights. Specifically, BEP trains a network using successive discriminative and generative phases such that an invertible model of the data is learned. Overall, weight asymmetry seems to allow greater transfer learning to the untrained task, while restricting training to a subset of weights worsens the transferability.

BEP is applicable to continuous-time neural models and temporally unrollable recurrent neural networks with asymmetric connection weights. The algorithm exploits a network's recurrent connections to learn the reciprocal generative transformation. It does so in a biologically plausible manner, using local computations for contrastive weight updates. It is consistent across a range of temporal scales, with fast neuronal dynamics, longer network trajectory navigation, synaptic plasticity and large-scale neural oscillations.

Essentially, bidirectional equilibrium propagation generalizes the contrastive learning rule for multiple "outputs", each governed by a clamping factor. The claim of the method is that with the right selection of clamping factors, to sample points in the activity space vector field, forward and backward mappings may be learned simultaneously. Experiments show a high degree of transfer learning when the network has asymmetric weights, whether the complimentary task direction weights are trained or not.

5.1 Future Work

As equilibrium propagation straddles the fields of machine learning, artificial neural networks, network science, dynamical systems and computational neuroscience, a variety of research directions building upon bidirectional equilibrium propagation involve the synthesis of two or more fields.

5.1.1 Bidirectional Learning

Bidirectional equilibrium propagation proposes a training methodology that not only copes with asymmetric forward and backward weights, but actually tries to exploit the representational capability this decoupling provides to learn a complimentary and invertible pair of mappings. While discriminative and generative models have been combined in artificial neural networks before, they have typically been in a serial arrangement. The canonical example of this is a Generative Adversarial Network [16], where a generative network is trained to generate samples from the distribution it is trying to model, while simultaneously a discriminative network is being trained to identify generated images from samples from the data distribution. In contrast, the generative-discriminative pair is inseparable. While this allows some degree of transfer learning across the forward/backward tasks, it has its disadvantages, namely the difficulty of learning in deeper networks. It shares a lack of convergence guarantees with other contrastive methods [42], since it does not follow the gradient of any objective function.

5.1.2 Finite Input Clamping

Equilibrium propagation uses the same type of neuronal computation at all times, addressing a biologically-motivated criticism of backpropagation. Yet, its phased operating modes require some external synchronization. Bidirectional equilibrium propagation relaxes this requirement by allowing bottom as well as top layers to be subjected to environmental cost functions. Nevertheless, there must still be some degree of synchronization between the clamping factors; i.e. one of α or β must be infinite. This constraint is, depending on the interpretation of the input layer, biologically motivated. For example, visual inputs from the retina to lateral geniculate nucleus (LGN) are not affected by further downstream processing; thus, considering retinal signals via the optic nerve to be the environment and the LGN to be the input layer would make an infinite α a reasonable assumption.

However, when modeling even just slightly higher areas of the visual stream, such as V1-V3 of the visual cortex, feedback or lateral connections on the hierarchy may change the input layer. Thus, considering the LGN to be the "environment" and its immediate projections to V1, V2 and V3 to be the input layer would necessitate a finite α. While this presents a more challenging optimization problem, addressing such variable clamping scenarios would be of interest from both a neuroscientific perspective, as well as a network science one.

5.1.3 Time-Varying Environment

Closely related to the problem of finite input clamping is that of time-varying input. With a finite clamping factor, the input is subject to change due to network feedback. With time-varying environmental input, the clamping factor may well be infinite, but the changing environmental input would force the input layer to change over time even without network feedback. The case of inputs varying on timescales much faster than synaptic weights is more biologically realistic.

In addition, recurrent neural networks are well-suited to processing time series data, due to the capability of network memory in sustained recurrent activity. A variety of artificial architectures have been successful in domains with sequences and temporal dependencies, such as long short-term memory [13]. Even echo state networks, randomly-connected pools of neurons with trained feedback weights, can learn to model time series data to some extent [21]. Thus, adapting a recurrent network such as BEP, or even vanilla EP, to time-varying inputs or "environments" would be a worthwhile area of study.

5.1.4 Biological Realism

A framework that claims to model biological neural networks should be able to replicate biological results at different spatial or temporal scales [46]. Whether this would be spiking neuron models, the ability to recreate network-level effects such as neural responses to certain stimuli, or to displaying layer-wise correspondence to biological activities, the ability to explain neural data would lend credibility to EP as a biological model.

Similarly, the ability to incorporate other biological processes at different spatial and temporal scales into the equilibrium propagation model would strengthen its case as a biologically-realistic model. In particular, intrinsic plasticity [43] would be a natural candidate. Intrinsic plasticity is the adaptation of the neuronal activation function based on the distribution of its inputs, by a variety of mechanisms generally occurring on a shorter

timescale than synaptic plasticity (i.e. weight modification). This serves to keep the neuron from operating in a saturated activation function regime. Thus, intrinsic plasticity avoids the dead neuron problem of Section 2.6.

Additionally, intrinsic plasticity has been used to improve the performance of echo-state networks [38], which are similar to equilibrium propagation networks due to their feedback connections. As deeper BEP networks would generally require a larger number of iterations to settle to equilibrium, take more training epochs to learn allowing longer timescale processes to take effect, and be more susceptible to vanishing error influence and dying neurons due to depth, intrinsic plasticity may be a useful computational tool as well as a biological feature.

Similarly, the viability of slower time-scale processes such as neural oscillations for modulating clamping factors could be investigated.

5.1.5 Deep Networks

While unidirectional EP has been shown to learn in deeper networks with more than one hidden layer [37], this proves to be a more difficult task for the bidirectional framework. Maintaining neural representations across layers would be the next computational challenge for bidirectional equilibrium propagation.

References

[1] Pierre Baldi and Fernando Pineda. Contrastive learning and neural oscillations. *Neural Computation*, 3(4):526–545, 1991.

[2] Horace B Barlow. Unsupervised learning. *Neural computation*, 1(3):295–311, 1989.

[3] Danielle S Bassett and Olaf Sporns. Network neuroscience. *Nature Neuroscience*, 20(3):353, 2017.

[4] Yoshua Bengio, Dong-Hyun Lee, Jorg Bornschein, Thomas Mesnard, and Zhouhan Lin. Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156*, 2015.

[5] Miguel A Carreira-Perpinan and Geoffrey E Hinton. On contrastive divergence learning. In *Aistats*, volume 10, pages 33–40. Citeseer, 2005.

[6] Paul Dean and John Porrill. The importance of Marrs three levels of analysis for understanding cerebellar function. *Computational Theories and their Implementation in the Brain: The legacy of David Marr*, page 79, 2016.

[7] Richard Durbin and David E Rumelhart. Product units: A computationally powerful and biologically plausible extension to backpropagation networks. *Neural Computation*, 1(1):133–142, 1989.

[8] Daniel Durstewitz, Jeremy K Seamans, and Terrence J Sejnowski. Neurocomputational models of working memory. *Nature Neuroscience*, 3(11s):1184, 2000.

[9] Karl Friston. The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11(2):127, 2010.

[10] Karl J Friston and Cathy J Price. Dynamic representations and generative models of brain function. *Brain Research Bulletin*, 54(3):275–285, 2001.

[11] Jonathan B Fritz, Mounya Elhilali, and Shihab A Shamma. Differential dynamic plasticity of A1 receptive fields during multiple spectral tasks. *Journal of Neuroscience*, 25(33):7623–7635, 2005.

[12] Nicholas Furl. Structural and effective connectivity reveals potential network-based influences on category-sensitive visual areas. *Frontiers in Human Neuroscience*, 9:253, 2015.

[13] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with LSTM. *Neural Computing*, 12(10):2451 – 2471, 2000.

[14] Aryn H Gittis and Sascha du Lac. Intrinsic and synaptic plasticity in the vestibular system. *Current Opinion in Neurobiology*, 16(4):385–390, 2006.

[15] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[17] Geoffrey E Hinton. Deterministic boltzmann learning performs steepest descent in weight-space. *Neural Computation*, 1(1):143–150, 1989.

[18] Gregor M. Hoerzer, Robert Legenstein, and Wolfgang Maass. Emergence of complex computational structures from chaotic neural networks through reward-modulated hebbian learning. *Cerebral Cortex*, 24(3):677–690, 2012.

[19] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

[20] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

[21] Herbert Jaeger. Adaptive nonlinear system identification with echo state networks. In *Advances in Neural Information Processing Systems*, pages 609–616, 2003.

[22] Andrzej Kasiński and Filip Ponulak. Comparison of supervised learning methods for spike time coding in spiking neural networks. *International Journal of Applied Mathematics and Computer Science*, 16:101–113, 2006.

[23] Christof Koch and Idan Segev. The role of single neurons in information processing. *Nature neuroscience*, 3(11s):1171, 2000.

[24] Hugo Larochelle, Michael Mandel, Razvan Pascanu, and Yoshua Bengio. Learning algorithms for the classification restricted Boltzmann machine. *Journal of Machine Learning Research*, 13(Mar):643–669, 2012.

[25] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[26] Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. Difference target propagation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 498–515. Springer, 2015.

[27] Qianli Liao, Joel Z Leibo, and Tomaso A Poggio. How important is weight symmetry in backpropagation? In *AAAI*, pages 1837–1844, 2016.

[28] Timothy P Lillicrap, Daniel Cownden, Douglas B Tweed, and Colin J Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7:13276, 2016.

[29] Henry Markram, Wulfram Gerstner, and Per Jesper Sjöström. Spike-timing-dependent plasticity: a comprehensive overview. *Frontiers in Synaptic Neuroscience*, 4:2, 2012.

[30] Pietro Mazzoni, Richard A Andersen, and Michael I Jordan. A more biologically plausible learning rule for neural networks. *Proceedings of the National Academy of Sciences*, 88(10):4433–4437, 1991.

[31] Arild Nøkland. Direct feedback alignment provides learning in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 1037–1045, 2016.

[32] Randall C O'Reilly. Six principles for biologically based computational models of cortical cognition. *Trends in Cognitive Sciences*, 2(11):455–462, 1998.

[33] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.

[34] Steven A Prescott and Terrence J Sejnowski. Spike-rate coding and spike-time coding are affected oppositely by different adaptation mechanisms. *Journal of Neuroscience*, 28(50):13649–13661, 2008.

[35] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[36] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533, 1986.

[37] Benjamin Scellier and Yoshua Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in Computational Neuroscience*, 11:24, 2017.

[38] Benjamin Schrauwen, Marion Wardermann, David Verstraeten, Jochen J Steil, and Dirk Stroobandt. Improving reservoirs using intrinsic plasticity. *Neurocomputing*, 71(7-9):1159–1171, 2008.

[39] Charles E Schroeder, Donald A Wilson, Thomas Radman, Helen Scharfman, and Peter Lakatos. Dynamics of active sensing and perceptual selection. *Current Opinion in Neurobiology*, 20(2):172–176, 2010.

[40] Sen Song, Kenneth D Miller, and Larry F Abbott. Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*, 3(9):919, 2000.

[41] David G Stork. Is backpropagation biologically plausible. In *International Joint Conference on Neural Networks*, volume 2, pages 241–246. IEEE Washington, DC, 1989.

[42] Ilya Sutskever and Tijmen Tieleman. On the convergence properties of contrastive divergence. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 789–795, 2010.

[43] Jochen Triesch. A gradient rule for the plasticity of a neurons intrinsic excitability. In *International Conference on Artificial Neural Networks*, pages 65–70. Springer, 2005.

[44] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[45] Xiaohui Xie and H Sebastian Seung. Equivalence of backpropagation and contrastive hebbian learning in a layered network. *Neural Computation*, 15(2):441–454, 2003.

[46] Daniel LK Yamins and James J DiCarlo. Using goal-driven deep learning models to understand sensory cortex. *Nature neuroscience*, 19(3):356, 2016.

[47] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.