University of Groningen

Master's Thesis

Analysis of Feedback Alignment

Author:
Pieter Jan Eilers
(s2381575)

Supervisors:
prof. dr. Michael Biehl
dr. Kerstin Bunte
Michiel Straat, Msc

September 9, 2021

rijksuniversiteit
groningen

Abstract

In the context of training neural networks, backpropagation of error with gradient descent is the most widely used method. With the backpropagation algorithm, the forward weights are reused in the feedback pass through the network. This process is not biologically plausible, as it requires neurons in hidden layers to know the synaptic weights of neurons in different layers. Recently, a new method has been suggested which shows that the feedback weights do not have to be identical and symmetrical to the forward weights. The weights used in the backward pass can be replaced by random feedback weights. The network will learn how to use these feedback weights effectively, essentially *learning how to learn*. In this thesis, we use on-line learning in student-teacher scenarios to compare the effectiveness of feedback alignment with the most commonly used backpropagation. We simulate several realizable, overrealizable and unrealizable scenarios for both shallow and deep networks. These networks use either the sigmoidal erf activation function or ReLU activation in the hidden neurons. Experiments show that feedback alignment can perform at least as efficiently and accurately as backpropagation in many scenarios. In simulations of deep over-parameterized student networks with both erf and ReLU activation, feedback alignment seems to have a systematic advantage in terms of earlier escape from learning plateau states where loss slows down significantly.

Contents

Common Abbreviations

- FA: Feedback Alignment
- BP: Backpropagation
- DFA: Direct Feedback Alignment
- ReLU : Rectified Linear Unit.
- erf: Gauss error function.
- SGD: Stochastic gradient descent
- 2-layer network: A feedforward neural network with one hidden layer.
- 3-layer network: A feedforward neural network with two hidden layers.

Common Notations

- $g(\cdot)$: Non-linear activation/transfer function, input and output are scalar values
- $\boldsymbol{g}(\cdot)$: Non-linear activation/transfer function, input and output are vectors with an equal number of components.
- $\mathcal{N}(0,1)$: Gaussian distribution with 0 mean and 1 variance.
- $\boldsymbol{\xi}^{\mu}$: Input example vector with i.i.d. components from $\mathcal{N}(0,1)$ at discrete timestep μ.
- N : Input dimension or system size.
- ϵ_g : Generalization error, the ability of the network to correctly classify novel data.
- $N-y-z$ network : A feedforward neural network with input dimension N, a hidden layer with y units and z output units
- $N-x-y-z$ network : A feedforward neural network with input dimension N, a hidden layer with x units, a second hidden layer with y units and z output units

1 Introduction

A neural network is a connected set of units or nodes known as neurons, loosely related to the neurons in a biological brain. Each connection can transmit a signal to another neuron and has a synaptic weight related to it. Training such a network simply means adapting the weights in a useful manner, such that when new data is presented, it is accurately responded to. Most neural networks are aggregated into several layers containing a number of hidden neurons. In the realm of supervised learning techniques, backpropagation is the most successful and commonly used technique in training deep neural networks. Its simplicity and overall performance makes it the obvious choice for training deep neural networks. However, this method requires that neurons must know the synaptic weights of other neurons, which is thought to be impossible in the brain [1, 2]. This makes backpropagation a biologically implausible method, as it requires neurons to send each other precise information about the synaptic weights of other hidden neurons.

A number of alternatives to backpropagation exist, including a variant of reinforcement learning, known as node perturbation [3]. This method applies small perturbations to the neuron activations in each layer. It then calculates potential weights by multiplying the normal inputs by the perturbations. However, this method also requires information that is not local to the neuron. There are alternatives inspired by biological processes, Boltzmann machine learning [4] and Contrastive Hebbian Learning [5], for example. More recently, another biologically plausible method known as difference target propagation (DTP) was introduced in [6], based on an older idea from [7]. Here, each layer is trained to reconstruct the layer below using backward inverse functions. These inverse functions are constructed using an auto-encoder at each layer. This is different from the widely used gradient descent, as activations, not gradients, are propagated back through the network. In this paper, a linear correction is used for the imperfectness of the auto-encoders, leading to results that are comparable to backpropagation.

The authors of [8] introduce a new learning strategy for neural networks, called feedback alignment (FA). In this paper, they observe that the weights used to back-propagate the gradient need not be identical and symmetrical to the forward weights. Fixed random feedback weights can be used instead, the network learns how to use these random weights to achieve convergence. Essentially, the network *learns how to learn*, which is an interesting albeit mystifying result. In order to use the feedback weights effectively, the forward weights start resembling the feedback weights, which allows for weight updates that resemble backpropagation. In other words, the forward weights align with the feedback weights, hence the term feedback alignment. This simpler method then uses mostly local information, as the weights of one layer do not need to know the weights of another layer. This training process more closely resembles the workings in a mammalian brain, making the method more biologically plausible compared to backpropagation.

In the original paper by Lillicrap et al. [8], several different learning scenarios are shown in which FA can match the performance of BP. A number of experiments are performed on artificially created data and on realistic datasets, including training a neural network to recognize handwritten digits using the MNIST dataset. Each of these experiments showed promising results. In [9], a different method is introduced, which propagates the error directly to the hidden layers, using random feedback connections. This method then uses almost entirely local information as it does not not require the error to travel backwards through reciprocal connections. This even simpler algorithm is able to reduce training error to zero in deep convolutional neural networks. The experiments in these two papers are all performed using either batch or mini-batch learning, also known as off-line learning. In this thesis, we focus on on-line learning scenarios, where the weights of a network are updated on each presentation of a single example. Moreover, every presentation of an example is new, never before seen by the network. This is different from off-line learning, where examples from a fixed training set are re-used.

We study a number of different student-teacher scenarios, where a neural network learns an unknown rule through stochastic gradient descent. Several Monte Carlo simulations are performed and stochastic on-line gradient descent is used to train these student networks, i.e. for the purpose of learning a regression scheme. This allows us to gain a better understanding of the dynamics of feedback alignment and it can provide a useful comparison with the performance of the more commonly used backpropagation. A framework such as this one borrows techniques from statistical physics that aggregate large physical systems in order to describe machine learning models that usually involve many adaptable weights. Instead of looking at each weights individually, we can aggregate these into so-called *order parameters*. This gives a more concise overview of the learning process, which makes large systems more interpretable. Additionally, a generalization error can be calculated either empirically or analytically, to determine the success of our training method.

As the weight parameters are iteratively updated in the learning process of a neural network, the loss decreases. However, in some settings, the loss might slow down significantly for a period of time, before rapidly increasing again. This *plateau phenomenon* can be seen in shallow and deep networks using backpropagation. The statistical mechanics of these phenomena have been studied in several situations [10, 11] and many methods have been proposed [12, 13] to break these plateaus at an earlier stage in the learning process. Using particular initializations of the order parameters in our student-teacher scenarios, we can "force" the training process to include long plateau states. By doing this, the effect of using FA on the length of plateau states can be researched.

The following section describes and explains the methods used. In section 3 and 4, the experiments are divided into two parts. Firstly, a number of results for different scenarios using shallow networks are shown for feedback alignment. Secondly, we take a look at deeper structures with 2 hidden layers for similar scenarios. At the conclusion of both experimental result sections, these results are discussed in more detail. Section 5 concludes the thesis, answering the following research questions:

- Firstly, we want to investigate the performance of feedback alignment in both shallow and deep networks to see if it is comparable to the performance of backpropagation. Moreover, if the performance changes depending on the differently learnable scenarios.

- Secondly, we would like to research the effect, if any, of feedback alignment on the length and frequency of unspecialized plateaus states. Furthermore, what happens to the alignment between the forward and the feedback weights in these fixed points.

- Thirdly, it would be interesting to know if feedback alignment requires the usage of a network with sigmoidal activation in the hidden units or if we can change it to an activation function of a different form like the ReLU function. The main differences lying in the simplicity and non-symmetry of ReLU compared to sigmoidal functions.

2 Methods

This section describes and explains the methods behind the experiments. On-line learning is generally executed using gradient descent with backpropagation. If we use feedback alignment, an adaptation has to be made in the backpropagation algorithm, replacing the forward weights by random feedback weights in the gradient calculations.

2.1 On-Line Learning of a Rule

In on-line learning using stochastic gradient descent(SGD), the gradient is calculated and weights are updated based on a single presentation of an example. This is different from off-line learning techniques like batch or mini-batch learning where a gradient is computed as an average over an entire dataset or a part of the dataset, respectively. In off-line learning, a fixed-size dataset, known as the training set is used to train the network. Another set, the test set, is used to calculate the accuracy of the network in correctly classifying novel data. In on-line learning, new examples $\boldsymbol{\xi}$ are presented in a stream, where each presentation leads to an adaptation of the student weights. In principle, one could still distinguish between the performance on previously seen examples and the test error for novel examples. However, because the weights are updated on every example and every example is never before seen by the network, only the generalization error is considered.

In order to define and model meaningful learning situations we resort to the consideration of student-teacher scenarios. In other words, the on-line learning of an unknown rule defined by either a two-layer network or a deeper feedforward neural network. For a two-layer neural network, the rule is defined by the structure of the network and its corresponding input-to-hidden synaptic weight matrix $\boldsymbol{C}$ and hidden-to-output weight vector $\boldsymbol{v}^*$. In a two-layer neural network with linear output, this set of input-to-hidden teacher weights is given by $\boldsymbol{C} \in \mathbb{R}^{M \times N}$. Here, N represents the input dimension and M represents the number of hidden units in the second layer. $\boldsymbol{C}_n$ is the n-th input branch and C_{mn} connects the n-th input value to the m-th hidden node. For initializations of $\boldsymbol{C}$ and input vector $\boldsymbol{\xi}^\mu$, each hidden unit n receives an an input $\boldsymbol{C}_n \cdot \boldsymbol{\xi}^\mu$. This can also be represented by the matrix-vector product $\boldsymbol{C}\boldsymbol{\xi}^\mu$, resulting in a vector of all input activations for the hidden units. In these hidden neurons, an non-linear activation function $\boldsymbol{g}(\cdot)$ is used, mapping the input to an activation. The total activation of all units

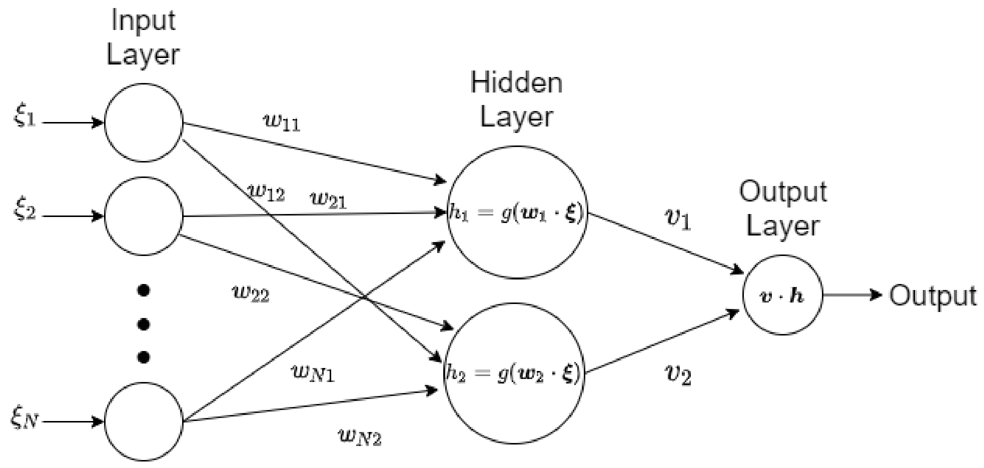

Input Layer
Hidden Layer
Output Layer
ξ_1
ξ_2
ξ_N
w_{11}
w_{12}
w_{21}
w_{22}
w_{N1}
w_{N2}
$h_1 = g(\boldsymbol{w}_1 \cdot \boldsymbol{\xi})$
$h_2 = g(\boldsymbol{w}_2 \cdot \boldsymbol{\xi})$
v_1
v_2
$\boldsymbol{v} \cdot \boldsymbol{h}$
Output

Figure 1: A two-layer network

would then be $\boldsymbol{g}(\boldsymbol{C\xi}^{\mu})$. This output activation is then either linearly or non-linearly combined in the output unit to produce the final state of the teacher. Figure 1 shows an example network with input data $\boldsymbol{\xi} \in \mathbb{R}^N$ and a single hidden layer. This network has N input units, $M = 2$ hidden units with a non-linear activation function, and a linear output unit. With a linear output unit, the final state is a linear combination of the hidden activations with the hidden-to-output weights $\boldsymbol{v}^* \in \mathbb{R}^M$. The state of a two-layer teacher network is

$$\tau(\boldsymbol{\xi}^{\mu}) = h\left(\sum_{n=1}^{M} v_n g\left(\boldsymbol{C}_n \cdot \boldsymbol{\xi}^{\mu}\right)\right). \tag{1}$$

With $\boldsymbol{C}_n$ denoting the N-dimensional teacher weight vector of the n-th input branch and v_n the weight connecting the n-th hidden neuron with the output unit. This formulation for the total state of the teacher can be altered to use only matrix-vector products and dot-products,

$$\tau(\boldsymbol{\xi}^{\mu}) = h\big(\boldsymbol{v}^{*}\cdot \boldsymbol{g}(\boldsymbol{C\xi}^{\mu})\big). \tag{2}$$

The output of $\boldsymbol{C\xi}^{\mu}$ is a vector, the activation function $\boldsymbol{g}(\cdot)$ applies to all vector components separately. Therefore, the output of $\boldsymbol{g}(\cdot)$ is also a vector with M components, the same number of components in $\boldsymbol{v}^*$, This allows the dot-product $\boldsymbol{v}^*{\cdot}\boldsymbol{g}(\cdot)$ between the two vectors, giving the final activation. In most of the following experiments, the activation function h in the output unit is be a linear combination of the hidden-to-output weights and the activations, allowing us to use the simpler notation:

$$\tau(\boldsymbol{\xi}^{\mu}) = \boldsymbol{v}^{*}\cdot \boldsymbol{g}(\boldsymbol{C\xi}^{\mu}). \tag{3}$$

A teacher network such as this can be extended to as many layers as one might desire. For example, when we have a 3-layer network with a linear output unit, the weight matrix $\boldsymbol{W}^* \in \mathbb{R}^{M_2 \times M_1}$ represents the weights connecting the first hidden layer to the second

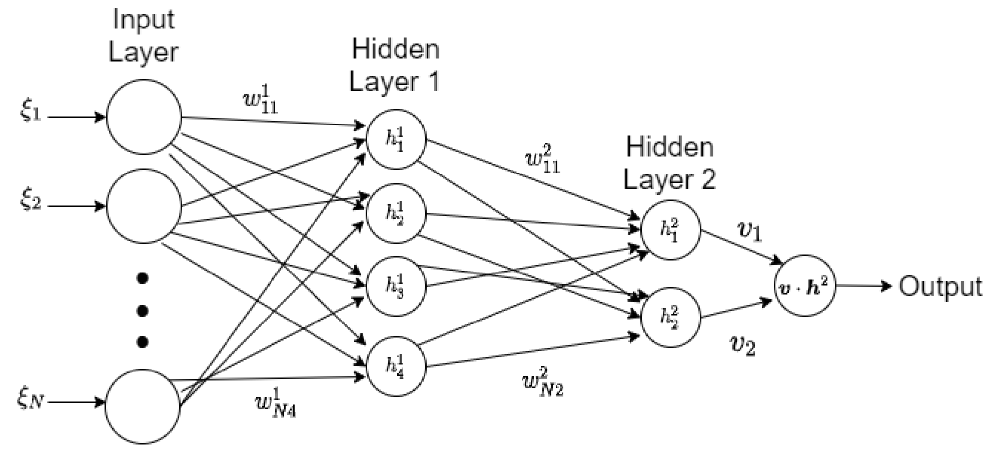


Figure 2: (a) A three-layer network with 2 hidden layers

one. Here M_1 and M_2 are the width of the first and second hidden layer, respectively. The final state would then be

$$\tau(\boldsymbol{\xi}^{\mu}) = \boldsymbol{v}^{*}\cdot \boldsymbol{h}(\boldsymbol{W}^{*}\boldsymbol{g}(\boldsymbol{C}\boldsymbol{\xi}^{\mu})). \tag{4}$$

Figure 2 shows an example network with 2 hidden layers with N input units, $M_1 = 4$ hidden units in the first layer, $M_2 = 2$ hidden units in the second layer and a linear output unit.

Now we can consider a second network, known as the student. This student network has an equal number of layers, however, the width of each hidden layer can differ. The student network may have fewer, equal or more units in the hidden layers. In our two-layer example, the first layer would have K hidden units with $\boldsymbol{J} \in \mathbb{R}^{K\times N}$. The student has no knowledge of the rule, as in other applications using a neural network, where a network learns a particular mapping which is unknown. The student output for input $\boldsymbol{\xi}^{\mu}$ is described as

$$\sigma(\boldsymbol{\xi}^{\mu}) = \boldsymbol{v}\cdot \boldsymbol{g}(\boldsymbol{J}\boldsymbol{\xi}^{\mu}). \tag{5}$$

The choice of a linear hidden-to-output transfer function does not restrict the capability of a student network to learn a complex rule. In a paper by Cybenko et al. [14] it is shown that two-layer networks with a linear output unit are known to be universal approximators. For deeper networks, the student state would look similar to the state for the teacher, with $\boldsymbol{C}$ replaced by $\boldsymbol{J}$. The hidden-to-hidden weights $\boldsymbol{W}$ and hidden-to-output weights $\boldsymbol{v}$ for the state of the student network are also different from $\boldsymbol{W}^{*}$ and $\boldsymbol{v}^{*}$ in the teacher. The weights $\boldsymbol{J}$ and $\boldsymbol{v}$ in a 2-layer network have to be adapted in the learning process. In a 3-layer network, $\boldsymbol{W}$ has to be adapted as well. The goal weights producing an output on input $\boldsymbol{\xi}$ that is as close as possible to the rule output.The training process of any neural network is generally guided by a loss function. For a network with real-valued output $y(\boldsymbol{\xi})$, based on examples $\{\boldsymbol{\xi}^{\mu} \in \mathbb{R}^{N}, \tau(\boldsymbol{\xi}^{\mu}) \in \mathbb{R}\}$, it would likely use quadratic deviation. In a student-teacher scenario, this is the quadratic deviation of the student network output with the rule output,

$$e^{\mu} = \frac{1}{2}\big(\sigma(\boldsymbol{\xi}^{\mu}) - \tau(\boldsymbol{\xi}^{\mu})\big)^2. \tag{6}$$

If a network is trained using off-line learning, we have a fixed set of training data which the student network uses to learn. The loss or error would then be an average over all training samples in a batch or mini-batch. This average error is then propagated back through the network in order to update the weights. A different set of examples, which the network has never seen called the test set, is used to calculate the network's ability to classify new data. In on-line learning, we do not have a fixed training set, as the weights are updated with each presented example. Because the examples are given in a stream, we only consider the generalization error. The average error made by the student is defined as the generalization error

$$\epsilon_g = \langle \epsilon \rangle_{\boldsymbol{\xi}}, \tag{7}$$

where the average, denoted by $\langle \cdot \rangle$ is taken over the distribution of random inputs. Evidently, when the student weights are updated such that the student output represents the teacher output as close as possible, e^{μ} will be reduced in the training process. After the forward pass, the loss e^{μ} is propagated back through the network, hence the term backpropagation. For a 2-layer network with a linear output unit, the weight vector $\boldsymbol{v}$, connecting the hidden layer to the output layer is reused together with the derivatives of the activation functions. The gradient with respect to the hidden-to-output weights is simply

$$\Delta \boldsymbol{v} = \frac{\partial}{\partial \boldsymbol{v}} e^{\mu} = \delta^{\mu} \boldsymbol{g}(\boldsymbol{J}\boldsymbol{\xi}^{\mu}), \tag{8}$$

where $\delta^{\mu} = \sigma(\boldsymbol{\xi}^{\mu}) - \tau(\boldsymbol{\xi}^{\mu})$.
We use this vector of gradients to update the input-to-hidden weights,

$$\boldsymbol{v}^{\mu+1} = \boldsymbol{v}^{\mu} - \frac{\eta}{K}\Delta \boldsymbol{v}. \tag{9}$$

Here, η represents the step size or *learning rate.* Note that the update is scaled by the width K of the hidden layer in the student network. Next, we can reuse δ^{μ} together with the derivative with respect to the input-to-hidden weights to calculate a new gradient vector $\boldsymbol{\delta}^{\mu}$ of component-wise results,

$$\boldsymbol{\delta}^{\mu} = \delta^{\mu} \boldsymbol{v} \odot \boldsymbol{g}'(\boldsymbol{J}\boldsymbol{\xi}^{\mu}). \tag{10}$$

The Hadamard product $\odot$ is the element-wise multiplication of two vectors. If $\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors with n elements, the Hadamard product is

$$(\boldsymbol{x} \odot \boldsymbol{y})_i = x_i y_i, \quad i = 1 \ldots n. \tag{11}$$

If FA is used in the update steps, $\boldsymbol{v}$ is replaced by a vector $\boldsymbol{b}$ with fixed random components,

$$\boldsymbol{\delta}^{\mu} = \delta^{\mu} \boldsymbol{b} \odot \boldsymbol{g}'(\boldsymbol{J}\boldsymbol{\xi}^{\mu}). \tag{12}$$

This is still only the partial gradient, the chain rule has to be applied another time. However, we have to specify the updates separately for each row $\boldsymbol{J}_i$ of the input-to-hidden student weights. Every component of the gradient vector $\boldsymbol{\delta}^\mu$ represents the cost attributable to that node. Consequently, the update of the input-to-hidden weights is

$$\boldsymbol{J}_i^{\mu+1} = \boldsymbol{J}_i^\mu + \frac{\eta}{N}\delta_i^\mu \boldsymbol{\xi}^\mu \text{ for } i = 1..K. \tag{13}$$

Here, the update is scaled by the system size N. For a deeper network, the $\boldsymbol{\delta}^\mu$ term is reused and the chain rule has to be applied several times. In a 3-layer network with 2 hidden layers, the gradient for the hidden-to-output weights is

$$\Delta \boldsymbol{v} = \frac{\partial}{\partial \boldsymbol{v}} e^\mu = \delta^\mu \boldsymbol{g}(\boldsymbol{x}_2), \tag{14}$$

with $\boldsymbol{g}(\boldsymbol{x}_2)$ being the output activation of the second hidden layer. The update of $\boldsymbol{v}$ is scaled by the width of the second hidden layer K_2. The component-wise gradients with respect to the hidden-to-hidden weights are

$$\boldsymbol{\delta}^\mu = \delta^\mu \boldsymbol{v} \odot \boldsymbol{g}'(\boldsymbol{x}_2). \tag{15}$$

Again, if we use FA, $\boldsymbol{v}$ is replaced by $\boldsymbol{b}$ and $\boldsymbol{\delta}^\mu$ becomes

$$\boldsymbol{\delta}^\mu = \delta^\mu \boldsymbol{b} \odot \boldsymbol{g}'(\boldsymbol{x}_2). \tag{16}$$

The update of $\boldsymbol{W}$ is

$$\boldsymbol{W}_i^{\mu+1} = \boldsymbol{W}_i^\mu + \frac{\eta}{K_1}\delta_i^\mu \boldsymbol{g}(\boldsymbol{x}_1), \tag{17}$$

where K_1 represent the number of units in the first hidden layer and $\boldsymbol{g}(\boldsymbol{x}_1)$ is the output activation of the first hidden layer. For the input-to-hidden weights, the component-wise gradients are

$$\boldsymbol{\delta}^\mu = \boldsymbol{W}^T \boldsymbol{\delta}^\mu \odot \boldsymbol{g}'(\boldsymbol{x}_1). \tag{18}$$

Here, the transpose of the hidden-to-hidden weight matrix $\boldsymbol{W}$ is used to propagate the error back through the network. When using FA, we can replace the symmetric forward weights $\boldsymbol{W}^T$ by a random matrix $\boldsymbol{B}$, resulting in

$$\boldsymbol{\delta}^\mu = \boldsymbol{B}\boldsymbol{\delta}^\mu \odot \boldsymbol{g}'(\boldsymbol{x}_1). \tag{19}$$

In the loss function landscape, BP provides a gradient in the direction of steepest descent, while FA does not. However, the network learns how to use $\boldsymbol{B}$ and $\boldsymbol{b}$, to provide a gradient in the general direction of BP. As long as the angle between the update of BP and FA is on average $\Delta\mathbf{h}_{FA} \measuredangle \Delta\mathbf{h}_{BP} < 90°$, the algorithm can take steps that resemble backpropagation. For a 2-layer network, this means that as long as on average $(e\boldsymbol{v}^T)(\boldsymbol{b}e) > 0$, the network can learn. The network does not need to have this property initially, as the angle between two feature vectors sampled from a distribution with 0 mean and 1 variance $\mathcal{N}(0,1)$ will be large. However, in the learning process, this angle will reduce over time as the alignment increases. This alignment implies that $\boldsymbol{b}$ begins to act like $\boldsymbol{v}$ and for deeper networks, $\boldsymbol{B}$ acts like $\boldsymbol{W}^T$ and the system can make updates that resemble BP. In [8], they indicate that feedback alignment, despite its simplicity, displays elements of second-order learning as feedback alignment encourages $\boldsymbol{W}$ to act like a local pseudo-inverse of $\boldsymbol{B}$. If $\boldsymbol{B}$ is exactly $\boldsymbol{W}^{+}$, the network would be performing Gauss-Newton optimization.

In the student-teacher scenarios, we have a target network with a fixed distribution of feature vectors sampled from $\mathcal{N}(0,1)$. This target network can then be learned by a student network through on-line learning, using backpropagation or feedback alignment. We assume the target network can be defined as a feedforward neural network. This can be either a 2-layer neural network with M units in the hidden layer, or another structure, such as a 3-layer network with 2 hidden layers. Both networks will have input dimension N. If a 2-layer neural network is used, the teacher has M hidden units, the student network has K hidden units, followed by a single output unit. A 3-layer network has two hidden layers. We structure these hidden layers, such that the first hidden layer in the target network has M hidden units and the following layer has $M/2$ hidden units. The student network would have K hidden units in the first layer and $K/2$ hidden units in the second layer. For instance, if $K = M = 4$, the 3-layer network for both student and teacher would look like the example network shown in Figure 2.

There are three different student-teacher scenarios with regards to hidden units. A *perfectly learnable* scenario has $K = M$, an equal amount of student and teacher units in every hidden layer. The two architectures match exactly, the student can entirely represent the rule. When this desired result is reached, the generalization error $\epsilon_g = 0$. Because we have an infinite stream of i.i.d. examples $\boldsymbol{\xi}$ in our on-line learning scenario, the perfect solution will always be reached, assuming the appropriate learning rate is chosen. In reality, the complexity of the target is often not known, hence the reason for studying cases with unequal complexity between the student and teacher network. An *overlearnable* scenario has $K > M$, the student network is more complex than the rule it is trying to learn. Over-parameterized networks are frequently used in deep learning contexts, where the number of parameters far exceeds the number of training examples. Generally, this comes with a risk of overfitting. However, recent understanding of the subject indicates that even over-parameterized networks can generalize well due to the implicit regularization in a gradient-based learning process [15]. Contrarily, an *unlearnable* scenario has $K < M$ and $\boldsymbol{C}_n \neq 0\ \forall n$, the student network lacks complexity in relation to the teacher network and can therefore not represent the rule perfectly. In other words, the network does not have a sufficient number of neurons to model a certain input-output mapping [16].

2.2 Sigmoidal Networks

There exist a number of sigmoidal activation functions, prominently used are tanh and the logistic function. The variety of sigmoidal functions satisfy the original purpose of an activation function, which is to simulate the firing rate of a biological neuron. All sigmoidal functions are characterized by its monotonic shape and bell-shaped first derivative. It is constrained by a pair of horizontal asymptotes as $x \to \pm\infty$. Because the theoretical concepts in this section are loosely based on previous studies [10, 17, 18], we consider the following sigmoidal activation function. On input x,

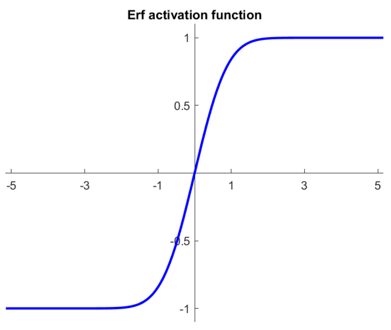
$$g(x) = \mathrm{erf}(x/\sqrt{2}). \tag{20}$$

In backpropagation using gradient descent, the derivatives of the activation function are used to propagate the error back through the network, The derivative of the erf function is

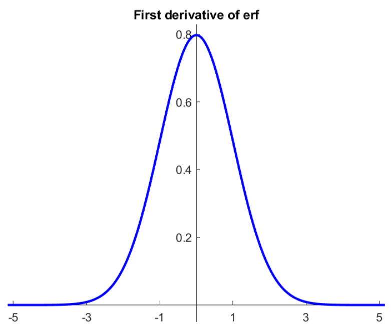
$$g'(x) = \sqrt{\frac{2}{\pi}} e^{-x^2/2}. \tag{21}$$

In Figure 3 we can see a representation of the erf function and its derivative, it is clear the symmetrical nature of the erf function. This is useful, individual student weights vectors $\boldsymbol{J}_i$, can have either positive or negative overlap with teacher vectors $\boldsymbol{C}_n$. At the end of training, students will either be specialized or anti-specialized to a teacher. If a student $\boldsymbol{J}_i$ is fully anti-specialized to a teacher $\boldsymbol{C}_n$, it is pointing in the exact opposite direction, the angle $\boldsymbol{J}_i \measuredangle \boldsymbol{C}_n = 180°$. With a positive overlap, the student is fully specialized at the end of training, $\boldsymbol{J}_i \measuredangle \boldsymbol{C}_n = 0°$. The reason this is possible is exactly due the symmetry of the erf function. If the overlap is negative, the activation function in the student network will produce the exact opposite value of the teacher. Combining this with the hidden-to-output weight, also opposites, the final output of the student and teacher network will be equal. Because of this symmetry, the derivative is Gaussian and feedback stage would have equal values for the activation derivatives. This concept is especially useful for FA, as the random feedback weights are fixed.

The bell-shaped derivative has a quickly decreasing gradient in both directions. In the training of neural network, this can form an issue over time. When the gradient becomes very small, the speed of training can slow down significantly. Small gradients are multiplied via the chain rule, resulting in even smaller updates for many layers. This problem known as the *vanishing gradient problem*, is one of the advantages of using another activation function with a different first derivative, explained in the following section.



(a)



(b)

Figure 3: (a) The sigmoidal erf activation function, (b) The Gaussian first derivative of erf.

2.3 ReLU Networks

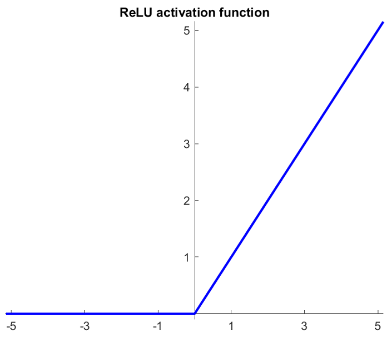
The *Rectified Linear Unit* is a non-linear activation function composed of two linear parts:

$$g(x) = x\theta(x),\ g'(x) = \theta(x), \tag{22}$$

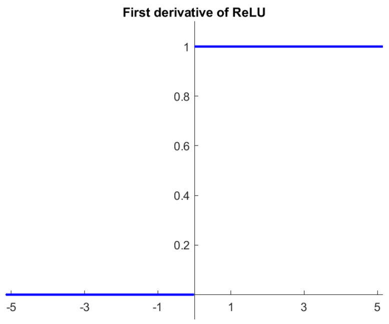
where $\theta(x)$ is the step function, defined as

$$\theta(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Immediately noticeable is the simplicity of the function, where the result of ReLU(x) on input x is either 0, when $x < 0$ or x when $x > 0$. This means the tangent is one, the derivative has only two values, which we can see in Figure 4. Because of this simple derivative, vanishing gradients cannot exist along paths of active hidden units. At ReLU$'(0)$, the function is discontinuous, a derivative does not exist. If this situation occurs in the training of a neural network with ReLU activation, a value of 0 or 1 for ReLU$'(0)$ can be chosen. For the experiments in the following sections using ReLU Networks, ReLU$'(0) = 0$. This choice seems rather trivial as ReLU$'(0)$ only occurs for a single value. However, in a paper by David Bertoin et al. [19], it is shown that the choice of ReLU$'(0)$ can impact the performance of backpropagation, especially when networks use limited numerical precision. Nevertheless, the general concensus implies that ReLU$'(0) = 0$ seems to be most robust. This derivative means that student weights are either updated with no scaling from the activation derivative, or they are not updated at all. For example, in Equation 12, $\boldsymbol{g}(\cdot)$ will be a vector of ones and zeros, and δ_i^μ will be 0 or the value of what is backpropagated. This is a significant difference from networks using the sigmoidal erf activation, where the magnitude of the updates varies more and is never exactly zero. An advantage of ReLU over sigmoidal activation is in its computational efficiency. Especially in deep networks, where many neurons have to compute the output and the gradient is evaluated as many times in the backpropagation steps. This computation takes more time with a complicated non-linear sigmoidal activation function like the erf function. Therefore, using ReLU activation in deep networks can reduce continuous training time. However, there are disadvantages as well, most notably, the gradient being zero when the input activation x_i upon presentation of example $\boldsymbol{\xi}^\mu$ is negative. If this is the case for all input vectors $\boldsymbol{\xi}^\mu$, the hidden neuron will never activate, since it is never updated. A solution to this problem, known as leaky ReLU has been suggested in [20] which has a small gradient for negative input activations. In the context of feedback alignment, there exists another downside to using ReLU activation. This is due to the non-symmetric nature of the ReLU function and the feedback weights being fixed. With the sigmoidal erf activation, student weights can either positively or negatively align with the teacher weights. The quadratic deviation $\frac{1}{2}\left(\sigma\left(\boldsymbol{\xi}^\mu\right) - \tau\left(\boldsymbol{\xi}^\mu\right)\right)^2$ would still be identical, as would the value for the derivatives of the activations. For the non-symmetric ReLU activation function this is not possible, which can form an issue when training ReLU networks using FA.



(a)



(b)

Figure 4: (a) ReLU activation: $x\theta(x)$, (b) The first derivative of the ReLU activation function: $\theta(x)$

2.4 Order Parameters

In this section, we go over some general statistical physics of on-line learning in a student-teacher scenario. Without going into elaborate detail, a number of useful properties are explained which can identify how a student learns the rule. For a more detailed theoretical analysis, [10, 11, 17, 21] are useful resources. In the previous section, a student-teacher scenario was defined for networks with 1 or 2 hidden layers. In this scenario, the teacher defines the rule to be learned by the student. Aside from the output, the student does not know anything about the rule. Among other benefits, a student-teacher scenario such as this allows for studying how the student learns the rule. In [17], learning dynamics for an on-line learning scenario are formulated in the thermodynamic limit. This means that the system is studied based on a stream of independent examples $\{\boldsymbol{\xi}^\mu, \tau(\boldsymbol{\xi}^\mu)\} \in \{\mathbb{R}^N, \mathbb{R}\}$ that are presented to the student and the system size $N \to \infty$. Input components of $\boldsymbol{\xi}^\mu$ are sampled from a distribution with zero mean and unit variance:

$$\xi_i = \mathcal{N}(0,1), \quad i = 1 \dots N. \tag{23}$$

We can use this simple density of the input and the definitions of the input activations

$$x_i = \boldsymbol{J}_i \cdot \boldsymbol{\xi}, \quad y_n = \boldsymbol{C}_n \cdot \boldsymbol{\xi}, \quad i = 1 \dots K, \quad n = 1 \dots M, \tag{24}$$

where x_i is the input activation to hidden student unit i and y_n is the input activation to hidden teacher unit n. Because the system is studied in the thermodynamic limit, by the Central Limit Theorem (CLT), these input activations are a multivariate Gaussian with a covariance matrix consisting of all covariances between pairs of input activations. Because of the assumed i.i.d. components of the input vectors $\boldsymbol{\xi} \in \mathbb{R}^N$,

$$\langle \xi_j \xi_l \rangle = \left\{ \begin{array}{ll} 1 & \text{for } j = l \\ 0 & \text{for } j \neq l \end{array} \right\} \tag{25}$$

the many degrees of freedom, i.e. the components of the adaptive vectors, can be characterized in terms of only very few quantities. The covariance between student input activation x_i and student input activation x_j is defined as

$$Q_{ik} = \sum_{j=1}^{N} \sum_{l=1}^{N} J_{ij} J_{kl} \langle \xi_j \xi_l \rangle = \sum_{j=1}^{N} J_{ij} J_{kj} = \boldsymbol{J}_i \cdot \boldsymbol{J}_k, \quad i,k = 1 \dots K. \tag{26}$$

Similarly, we can define the covariance between student-teacher input activations and the covariance between teacher-teacher input activations as follows:

$$R_{im} = \boldsymbol{J}_i \cdot \boldsymbol{C}_m, \quad T_{mn} = \boldsymbol{C}_m \cdot \boldsymbol{C}_n, \quad i = 1 \dots K, \quad m, n = 1 \dots M. \tag{27}$$

Note that that the rule constants T_{mn} are fixed, as they define the rule, i.e. the parameters of the teacher network. If the rule covariances are isotropically initialized, then $T_{mn} = 0$ for $m \neq n$. These covariances or overlaps, are the so-called *order parameters*, able to aggregate many parameters into a descriptive summary. Such parameters are especially helpful when studying on-line rule-learning scenarios in 2-layer networks. In the thermodynamic limit $N \to \infty$, the order parameters are self-averaging, which allows for exact continuous learning dynamics to be formulated. Using the order parameters, an exact analytical solution for the generalization error ϵ_g can be formulated as an expected average over the simple input density. In early stages of learning, a unstable fixed point in the learning dynamics may be reached where the loss decrease slows down significantly. In such a plateau state, the corresponding student-teacher overlaps are either close to identical or close to exactly opposite. When such a symmetry is broken, the students will specialize and the loss will rapidly decrease again. We can display the evolution of the order parameters after training to observe when a student $\boldsymbol{J}_i$ specializes to a certain teacher vector $\boldsymbol{B}_n$, also showing when and how the system might break out of a plateau state. In networks with deeper layers, the order parameters are still a useful tool. However, they can not be used in the calculation of an analytical generalization error, as there exists no analytical solution for deeper networks. One could design order parameters like this for other layers. This is rather unnecessary, as the other layer weights have significantly lower input dimension, there is no need to aggregate many degrees of freedom into very few quantities.

2.5 Generalization Error

During the on-line learning process, the success of learning can be measured in terms of the generalization error ϵ_g. This quantity, also known as the test error in many fields of machine learning, represents the accuracy of the network correctly responding to new input data. In the thermodynamic limit $N \to \infty$, we can rely on the simple density of the input $\boldsymbol{\xi}$ with i.i.d. components sampled from $\mathcal{N}(0,1)$ to define the generalization error for a 2-layer network as the expected prediction error on the presentation of novel examples:

$$\epsilon_g(\boldsymbol{J}, \boldsymbol{v}) = \langle \epsilon(\boldsymbol{J}, \boldsymbol{v}) \rangle_{\boldsymbol{\xi}}. \tag{28}$$

An expression for the generalization error can be derived using the order parameters, rule constants and hidden-to-output weights $\boldsymbol{v}$. For a 2-layer network with erf activation, this equates to

$$\begin{aligned}\epsilon_g =& \frac{1}{\pi}\Bigg[\sum_{i,j=1}^{K} v_i v_j \sin^{-1}\left(\frac{Q_{ij}}{\sqrt{1+Q_{ii}}\sqrt{1+Q_{jj}}}\right) \\ &- 2\sum_{i=1}^{K}\sum_{m=1}^{M} v_i v_m^* \sin^{-1}\left(\frac{R_{im}}{\sqrt{1+Q_{ii}}\sqrt{1+T_{nn}}}\right) \\ &+ \sum_{m,n=1}^{M} v_m^* v_n^* \sin^{-1}\left(\frac{T_{mn}}{\sqrt{1+T_{mm}}\sqrt{1+T_{nn}}}\right)\Bigg],\end{aligned} \tag{29}$$

here v_i represents the i-th hidden-to-output student weight and v_m^* the m-th hidden-to-output teacher weight. The original derivation of this analytical solution for Soft Committee Machines is from [18] and second layer weights were introduced in [10]. When the rule parameters are isotropic and have unit norm, $T_{mm} = 1$ and $T_{mn} = 0$ for $m \neq n$, this simplifies to

$$\begin{aligned}\epsilon_g =& \frac{1}{\pi}\Bigg[\sum_{i,j=1}^{K} v_i v_j sin^{-1}\left(\frac{Q_{ij}}{\sqrt{1+Q_{ii}}\sqrt{1+Q_{jj}}}\right) \\ &- 2\sum_{i=1}^{K}\sum_{m=1}^{M} v_i v_m^* sin^{-1}\left(\frac{R_{im}}{\sqrt{1+Q_{ii}}\sqrt{2}}\right) \\ &+ \sum_{m=1}^{M} sin^{-1}\left(\frac{(v_m^*)^2}{4}\right)\Bigg].\end{aligned} \tag{30}$$

For a 2-layer network with ReLU activation, an analytical solution for the generalization error is also known. In [11], the formula for ϵ_g is derived for a ReLU SCM architecture which has fixed and equal hidden-to-output weights $\boldsymbol{v}$. Including these weights is trivial, as they can be linearly combined with the order parameters and rule constants.

$$\begin{aligned}\epsilon_g =& \frac{1}{2}\Bigg[\sum_{i,j=1}^{K} v_i v_j \left(\frac{Q_{ij}}{4} + \frac{\sqrt{Q_{ii}Q_{jj} - Q_{ij}^2}}{2\pi} + \frac{Q_{ij}\sin^{-1}\left(\frac{Q_{ij}}{\sqrt{Q_{ii}Q_{jj}}}\right)}{2\pi}\right) \\ &- 2\sum_{i=1}^{K}\sum_{m=1}^{M} v_i v_m^* \left(\frac{R_{im}}{4} + \frac{\sqrt{Q_{ii}T_{mm} - R_{im}^2}}{2\pi} + \frac{R_{im} sin^{-1}\left(\frac{R_{im}}{\sqrt{Q_{ii}T_{mm}}}\right)}{2\pi}\right) \\ &+ \sum_{m,n=1}^{M} v_m^* v_n^* \left(\frac{T_{mn}}{4} + \frac{\sqrt{T_{mm}T_{nn} - T_{mn}^2}}{2\pi} + \frac{T_{mn} sin^{-1}\left(\frac{T_{mn}}{\sqrt{T_{mm}T_{nn}}}\right)}{2\pi}\right)\Bigg].\end{aligned} \tag{31}$$

Again this can be simplified with isotropically initialized teachers that have unit norm, to

$$\begin{aligned}\epsilon_g =& \frac{1}{2}\Bigg[\sum_{i,j=1}^{K} v_i v_j \left(\frac{Q_{ij}}{4} + \frac{\sqrt{Q_{ii}Q_{jj} - Q_{ij}^2}}{2\pi} + \frac{Q_{ij}\sin^{-1}\left(\frac{Q_{ij}}{\sqrt{Q_{ii}Q_{jj}}}\right)}{2\pi}\right) \\ & - 2\sum_{i=1}^{K}\sum_{m=1}^{M} v_i v_m^* \left(\frac{R_{im}}{4} + \frac{\sqrt{Q_{ii} - R_{im}^2}}{2\pi} + \frac{R_{im} sin^{-1}\left(\frac{R_{im}}{\sqrt{Q_{ii}}}\right)}{2\pi}\right) \\ & + \sum_{m=1}^{M}\left(\frac{(1-\pi)(v_m^*)^2}{2\pi}\right) + \sum_{n=1}^{M}\frac{v_m^* v_n^*}{2\pi}\Bigg].\end{aligned} \tag{32}$$

For 2-layer networks with a non-linear output unit, or networks with 3 or more layers, there exists no analytical solution. An empirical estimate has to be relied on as a measure of overall performance. For example, we can take the quadratic deviation of student and teacher output on random input $\boldsymbol{\xi}^\mu$, averaged over N_s test samples,

$$\epsilon_g = \frac{1}{N_s}\sum_{\mu=1}^{N_s}\frac{1}{2}(\sigma(\boldsymbol{\xi}^\mu) - \tau(\boldsymbol{\xi}^\mu))^2. \tag{33}$$

For each presentation of a test example, the weights do not get updated, as that would skew the final result of ϵ_g. Of course, doing these calculations for every presented example would be extremely time consuming, especially when learning convergence requires the presentation of many examples. Generally, we calculate the order parameters and the generalization error after one or multiple presentations of at least N examples.

2.6 Role of the Learning Rate

The learning rate is a common theme in the realm of machine learning and it can be a challenge to find the η giving optimal performance. If the learning rate is too high, the process will diverge and a solution will not be reached. If the learning rate is too small, the learning process will likely converge but it can take a large amount of time. Generally, η will be lower when a rule is learnt using ReLU networks, compared to when erf networks are used. The reason for this lies in the fact that the gradient in the update steps for the ReLU activation function is generally higher. This increases the magnitude of the actual updates, essentially increasing the learning rate, allowing for a lower η to be considered. In the context of online learning with stochastic gradient descent, a larger η can be used in the early stages of learning. Once the error function is close to a global or local minimum, a small η is necessary to let the process converge. If η is too large, the process will oscillate around this minimum indefinitely. For this reason, many adaptive learning rate methods exist. Common methods include time-based decay and exponential decay. In the following experiments, we will stick to an empirically chosen small fixed learning rate which ensures convergence in solvable scenarios.

3 Results for Shallow Networks

This section describes several experiments for a number of different setups using shallow networks. A shallow network has one hidden layer. The hidden neurons in these networks use a non-linear activation function, either erf or ReLU. All experiments represent the on-line learning of a rule through stochastic gradient descent. Networks are trained using a Monte Carlo setup, which means all input data $\boldsymbol{\xi} \in \mathbb{R}^N$ is randomly generated from an independent normal distribution $\mathcal{N}(0,1)$ and given in a stream. We compare the performance of backpropagation (BP) with that of feedback alignment (FA), using metrics as the generalization error and order parameters, explained in the previous section. Input-to-hidden weights are sampled from a normal distribution $\mathcal{N}(0,1)$ and scaled by $\sqrt{N}$. This scaling reduces the magnitude of the variance for the pre-activations. Among other advantages, this reduces the chance of vanishing gradient issues when the hidden layer activation is too high. A generalized method of Gram-Schmidt orthogonalization is used to initialize the student and teacher weights $\boldsymbol{J}$ and $\boldsymbol{B}$ with preset overlap,

$$R_{im} = 0, \quad Q_{ii} = 0.5, \quad Q_{ik} = 0.49 \text{ for } i \neq k \tag{34}$$

for $i, k = 1 \dots K$ and $m = 1 \dots M$. The rule constants are initialized isotropically with unit norm, $T_{mm} = 1$ and $T_{mn} = 0$ for $m \neq n$. The student weights $\boldsymbol{J}$ are initialized such that the individual vectors $\boldsymbol{J}_i$ have high overlap between them. This initial high overlap increases the symmetry between students, which in turn increases the difficulty of a student vector $\boldsymbol{J}_i$ specializing to a teacher vector $\boldsymbol{C}_i$, often resulting in longer plateau states. Furthermore, the overlaps between $\boldsymbol{J}$ and $\boldsymbol{C}$ are initialized, such that all students are orthogonal to all teachers, students have no initial overlap with any teacher. In situations where the hidden-to-output weights are fixed and identical, as in [17], initializing the student with zero overlap would be unwise as the system can get stuck in a fixed point indefinitely. The student network will not be able to specialize. However, with variable hidden-to-output weights $\boldsymbol{v}$ randomly sampled from $\mathcal{N}(0,1)$ in our experiments, students will naturally have some initial overlap with the teacher weights. Other feature vectors, $\boldsymbol{v}^*$ and $\boldsymbol{b}$ are sampled from $\mathcal{N}(0,1)$ as well. In the graphs shown below, the x-axis scale is generally $\alpha = \mu/N$, the total timesteps divided by the input dimension. This can be seen as one run trough a dataset with equal dimension as input examples.

3.1 Sigmoidal Networks

We begin our experiments with the analysis of FA in 2-layer sigmoidal erf networks, comparing its effectiveness to BP. Here, we are using a 500-4-1 student network in a perfectly learnable scenario. Thus the input layer has dimension $N = 500$, the hidden layer has $K = M = 4$ units in both student and teacher and there is one linear output unit. When using FA, the hidden-to-output weight vector $\boldsymbol{v}$ used in the backpropagation step, is replaced by a vector $\boldsymbol{b}$ with random components. In Figure 5, we can see the generalization error for a single simulation with identical initial conditions for BP and FA. The learning rate is fixed at $\eta = 0.1$. We can see that even for identical initial conditions, performance can differ quite substantially. Both methods have multiple plateau states of different length and position. After breaking these plateau states, both methods asymptotically reduce the error to zero. Figure 7 showcases the order parameters R_{im} for BP in blue and FA in red. Each graphs shown 4 lines for both FA and BP, representing the 4 students. At the end of training, three of these lines will go to 0, while the remaining one will go to either 1 or -1. This represents a student being either fully specialized or anti-specialized. This specialization depends on the signs of $\boldsymbol{v}^*$, the signs of $\boldsymbol{v}$ for BP and the signs of $\boldsymbol{b}$ for FA. Because the random feedback vector $\boldsymbol{b}$ is fixed, the alignment is also fixed to that sign. For BP, the signs of the forward weights $\boldsymbol{v}$ could flip over the time of training, which would invert the overlap with the teacher. We are also interested in the overall performance, as an average over multiple simulations. The adjacent plot shows this average, where the evolution of ϵ_g is logarithmically scaled. This graph represents an average over $n = 10$ simulations. Because of the variance that can occur between the two methods, the early stages of ϵ_g can be quite different between the two methods. Large plateau states in one simulation can dominate the average, especially on a logarithmic scale. However, after all unspecialized states have been broken, the final exponential decrease of ϵ_g is very similar, as the slope is almost identical. The alignment plot in Figure 6a shows the inverse cosine alignment between $\boldsymbol{v}$ and $\boldsymbol{b}$ in degrees,

$$\theta = cos^{-1}\frac{(\boldsymbol{b} \cdot \boldsymbol{v})}{(||\boldsymbol{b}||\ ||\boldsymbol{v}||)}. \tag{35}$$

This alignment represents the angle between the hidden unit changes prescribed by FA and those prescribed by BP, $\Delta\mathbf{h}_{FA}\measuredangle\Delta\mathbf{h}_{BP}$. Initially, this angle is large, for sufficiently large feedback vector, this angle would be close to 90 degrees. However, for $\boldsymbol{b}$ used here with a shorter length, the initial angle can be smaller or larger. Over time, the angle decreases as the network learns how to use the feedback weights effectively. This decrease in angle indicates that the algorithm takes steps that are closer to backpropagation. In the alignment plot, after an initial decrease and fluctuation, the alignment settles at a final angle. The network has found an optimal alignment with $\boldsymbol{b}$ for which the error can reduce to zero. In Figure 6b, this alignment is overlayed on the evolution of ϵ_g for FA from Figure 5a. Here, the values on the y-axis only correspond to the values of the generalization error. The alignment overlayed on top is shown only for purposes of comparing the fluctuations in the alignment with the plateau states in the loss decrease. This overlaying is useful to observe the behaviour of the alignment between the updates made BP and those made by FA when the learning has reached a loss plateau. Here, for all three plateau states, the updates made by BP and those made by FA de-align, as the angle increases. When this state is broken, the updates swiftly re-align as the angle decreases.

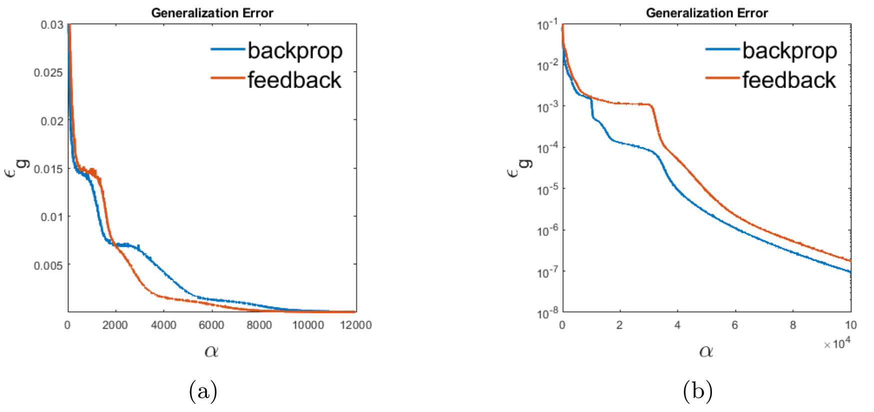


Figure 5: (a) Evolution of the generalization error ϵ_g for FA and BP in a sigmoidal 500-4-1 network with erf activation. The initial conditions are identical for both methods, (b) Plot of the average ϵ_g on a logarithmic scale for $n = 10$ simulations with different random initializations.

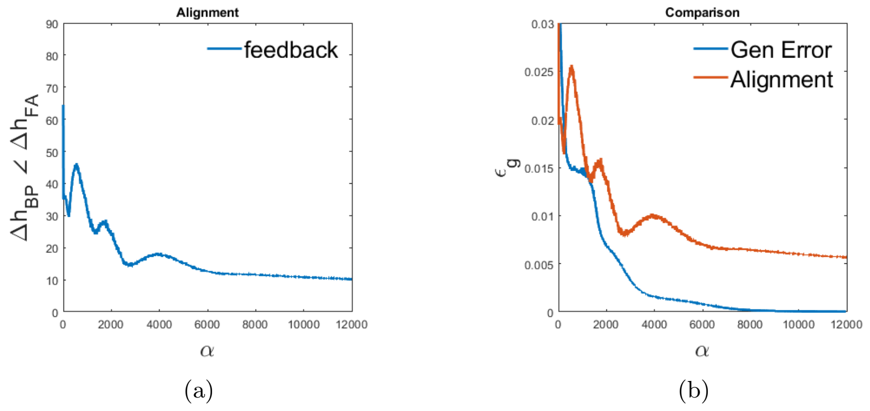


Figure 6: (a) Inverse cosine alignment between the $\boldsymbol{b}$ and $\boldsymbol{v}$. This alignment represents the angle between the hidden unit changes prescribed by BP with those prescribed by BP $\Delta\mathbf{h}_{FA}\measuredangle\Delta\mathbf{h}_{BP}$. (b) The inverse cosine alignment overlayed on ϵ_g for FA of Figure 5a
.

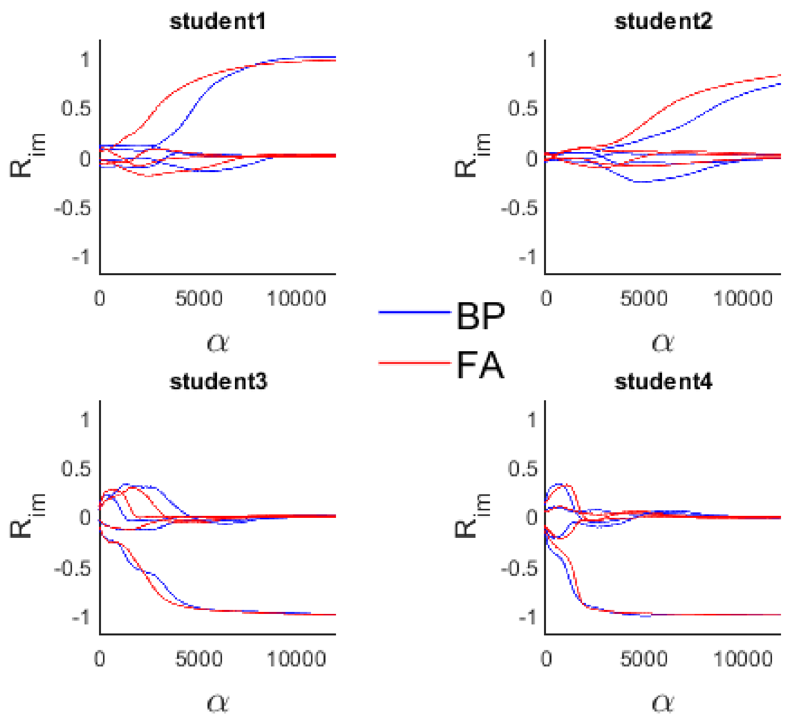


Figure 7: Order parameters R_{im} for FA and BP in a 500-4-1 erf network. Both methods have identical initial conditions. BP is shown in blue, FA in red.

3.1.1 Non-Linear Output

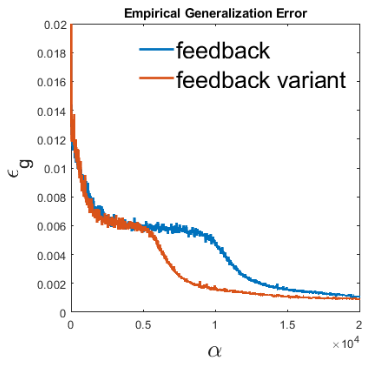
The output unit can also be non-linear with an erf activation function, mapping the activation in the output unit to a value between -1 and 1. If we use a network like this, the gradient for the hidden-to-output weights becomes

$$\frac{\partial}{\partial \boldsymbol{v}} e^{\mu} = (\sigma(\boldsymbol{\xi}^{\mu}) \ - \tau(\boldsymbol{\xi}^{\mu}))\ h'(\boldsymbol{v} \cdot \boldsymbol{g}(\boldsymbol{J}\boldsymbol{\xi}^{\mu}))\boldsymbol{g}(\boldsymbol{J}\boldsymbol{\xi}^{\mu}). \tag{36}$$

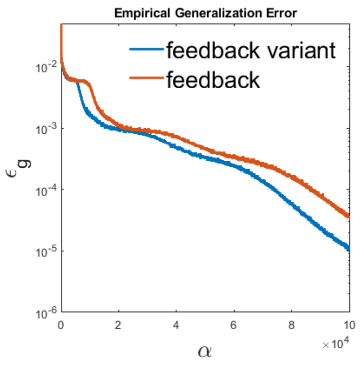
One could argue that this update is not biologically plausible, as it requires the individual nodes of the hidden layer to know the weights of the other nodes in the hidden layer. When a node i with synaptic weight v_i is updated, it requires knowledge of all other components in $\boldsymbol{v}$. We could replace the $\boldsymbol{v}$ by the random feedback vector $\boldsymbol{b}$ here as well. This changes the gradient to

$$\frac{\partial}{\partial \boldsymbol{v}} e^{\mu} = (\sigma(\boldsymbol{\xi}^{\mu}) - \tau(\boldsymbol{\xi}^{\mu}))\ h'(\boldsymbol{b} \cdot \boldsymbol{g}(\boldsymbol{J}\boldsymbol{\xi}^{\mu}))\boldsymbol{g}(\boldsymbol{J}\boldsymbol{\xi}^{\mu}). \tag{37}$$

If we do this, the value for the derivative h' changes, depending on the result of $\boldsymbol{b}\cdot g(\boldsymbol{J}\boldsymbol{\xi}^{\mu})$). Consequently, the magnitude of the gradient would increase or decrease. Figure 8 shows the evolution of ϵ_g for both these FA options. Because there currently exists no analytical solution for ϵ_g in a 2-layer network with a non-linear output unit, we calculate an empirical estimate, using Equation (33). This is reflected in the graph, as the resulting plots lack smoothness compared to the results for an analytical solution of ϵ_g. The linear plot on the left shows that this different variant of FA works as well as the normal variation, even breaking the plateau state earlier. For the right plot, we have increased α and ϵ_g is scaled logarithmically. Aside from a slightly earlier escape from the unspecialized state, convergence to zero ϵ_g looks almost identical for both variants of FA. Even the angle of alignment over time is close to identical, as shown in Figure 9.



(a)



(b)

Figure 8: (a) Evolution of an empirical estimate ϵ_g versus α for FA and the different variant of FA for a 500-4-1 student and teacher network with erf activation. Both methods have identical initial conditions (b) The same estimated ϵ_g on a logarithmic scale with increased α.

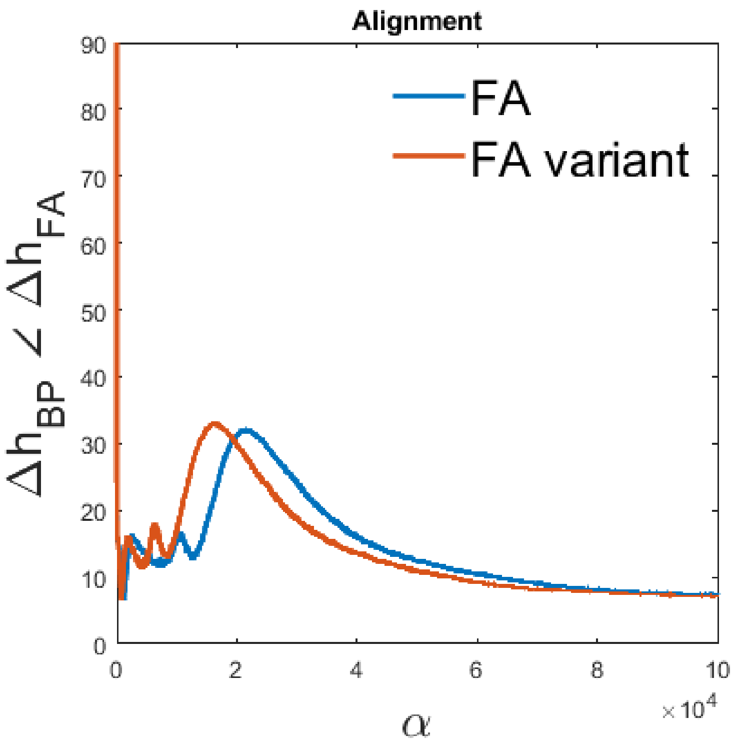
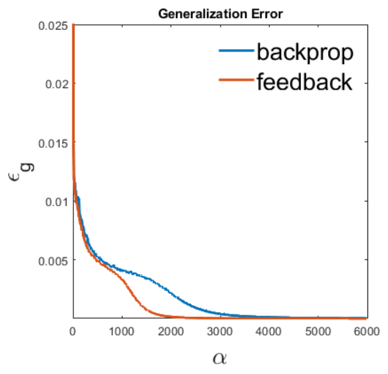


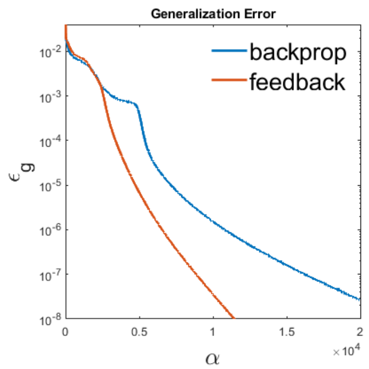
Figure 9: Evolution of the inverse cosine alignment representing the angle between the updates made by FA and those made by BP in a 500-4-1 student and teacher network. This angle is shown for FA and the different variant of FA which changes the activation derivative in the gradient.

3.1.2 Overrealizable, $K > M$

The same student network setup is used in the following experiment, however, we reduce the complexity of the teacher network. Here, $N = 500$, $K = 4$ and $M = 2$, making it an overrealizable scenario. In general, this means that either students will share specialization. In this case, the absolute value for the order parameters R_{im} for multiple students specializing to the same teacher should add up to 1. More likely, some students will be seen as a redundancy and these will be phased out, reducing either the corresponding $\boldsymbol{J}_i$ or $\boldsymbol{v}_i$ to zero, or both. In Figure 10a we see the evolution of ϵ_g for both BP and FA. The resulting graph represents a single simulation with $\eta = 0.1$ and identical initial conditions for both methods. Here, we can see that FA breaks out of the short plateau state earlier and ϵ_g is asymptotically reduced to zero. Figure 11 shows the order parameters R_{im} corresponding with the ϵ_g graph for BP in blue and FA in red. Here we can see the adaptation the student network makes when it has increased complexity compared to the teacher. Two students specialize to two teachers, while the other two are phased out. For both methods, the graph would indicate that two students that should be phased out still have some overlap with the teacher. This is due to the fact that the hidden-to-output weights $\boldsymbol{v}$ will also set the corresponding values to 0, which might happen before the $\boldsymbol{J}_i$ are made completely redundant. In this example experiment, the hidden-to-output teacher weight vector $\boldsymbol{v}^* = [-0.5003, 0.6596]$. After training, the hidden-to-output student weight vector $\boldsymbol{v} = [0, 0, -0.6596, 0.5003]$ for FA and $\boldsymbol{v} = [0, 0, -0.6596, 0.5004]$ for BP. Figure 12a displays the evolution of the alignment between the hidden-unit updates prescribed by BP and those prescribed by FA. After an initial fluctuation, the angle follows a slight increase until settling at the final unchanging angle. The adjacent graph shows the inverse cosine alignment overlayed on the ϵ_g graph for FA from Figure 10a. From this graph, it is shown that the increase of the angle between $\boldsymbol{v}$ and $\boldsymbol{b}$ occurs at the small plateau state. To get a better idea of the general performance of FA in an overrealizable setting, we again take an average over multiple simulations and show the evolution of ϵ_g on a logarithmic scale. The resulting graph is shown in Figure 10b as an average over $n = 10$ simulations for both methods. Again, FA is shown to reach convergence slightly earlier due to earlier escapes from loss plateaus.



(a)



(b)

Figure 10: (a) Evolution of ϵ_g for a 500-2-1 teacher network and a 500-4-1 student network with erf activation. The results are for a single simulation with identical initial conditions for both FA and BP. (b) Evolution of ϵ_g on a logarithmic scale. The results are averaged over $n = 10$ simulations.

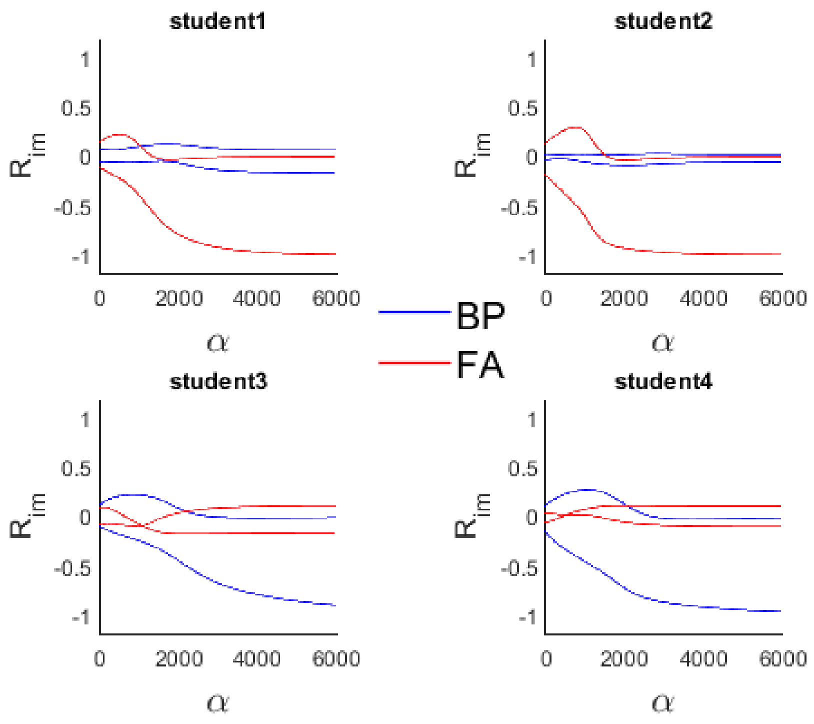


Figure 11: Order parameters R_{im} for a 500-2-1 teacher network and a 500-4-1 student network with erf activation. FA is shown in red, BP is shown in blue.

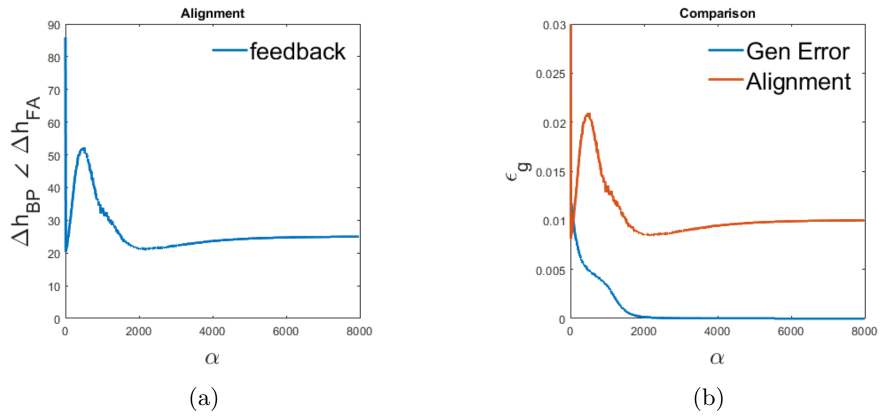
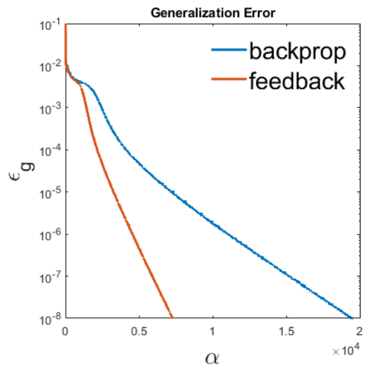
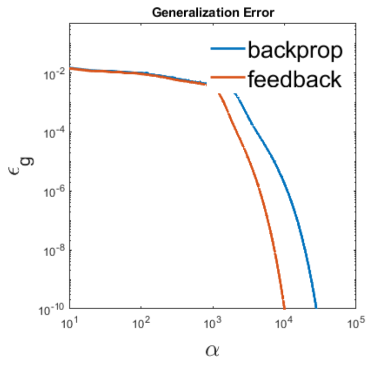


Figure 12: (a) Inverse cosine alignment angle between the updates made by BP and those made by FA in an overrealizable scenario for a 500-4-1 student and a 500-2-1 teacher network with erf activation. (b) The inverse cosine alignment overlayed on the evolution of ϵ_g for FA from Figure 10a

In a paper by Frederieke Richert et al. [15], it is shown that for identical sigmoidal activation function $\mathrm{erf}(x/\sqrt{2})$ and overrealizable scenarios, the generalization error changes from exponential decrease to a different power-law rate of convergence, with $\epsilon_g \propto 1/\alpha^2$. The setup in this paper is different however, an SCM is used which has fixed unit hidden-to-output weights. Moreover, a particular normalization is used which allows all student vectors to learn, all students are seen as relevant by the system and no student will be set to zero. In Figure 13 we can see the evolution of ϵ_g from Figure 10a on a single and a double logarithmic scale. Here we can see that when ϵ_g decreases, this happens exponentially. The lines in the single logarithmic plot, after the plateau states is broken, go from a steep straight line to a somewhat less steep line. This indicates that the decrease of ϵ_g stays exponential, albeit slightly slower from a certain point on. If there was a power-law convergence, ϵ_g in the single logarithmic graph would be asymptotically bound on the x-axis. The tail of the double logarithmic graph would have a straight line for ϵ_g, as the slope would be proportional to $\frac{1}{\alpha}$. In Figure 13b, it is clear that the tail of the graph is curved, asymptotically bound on the y-axis.
[15].



(a)



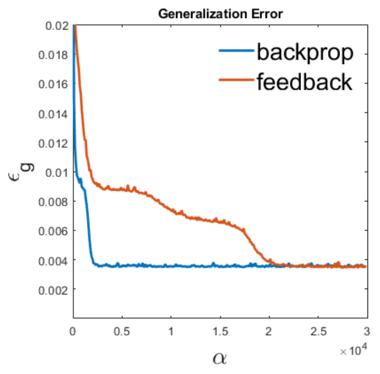
(b)

Figure 13: (a) Evolution ofϵ_g on a logarithmic scale for both FA and BP for a 500-2-1 teacher network and a 500-4-1 student network, (b) Evolution of ϵ_g on a double logarithmic scale

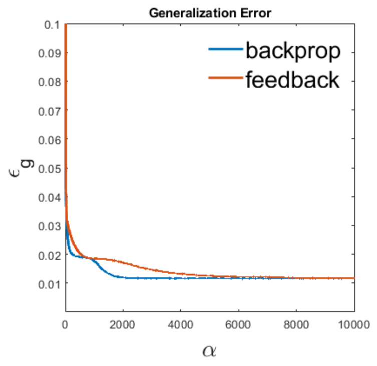
3.1.3 Unrealizable, $K < M$

We are also interested in the opposite situation in relation to the complexity of student and teacher, an unrealizable scenario with $K < M$. The student network lacks complexity compared to the teacher. The previous scenario is flipped, now $K = 2$ and $M = 4$. In an unrealizable scenario, it is impossible to achieve zero generalization error as the student cannot fully realize the rule. The student cannot specialize to all teacher weight vectors, so a balance has to be found for which a global minimum can be reached. We can see this in Figure 14a, showing the generalization error for both FA and BP. This graph shows the results for a single simulation with $\eta = 0.1$, where the initial conditions are identical for both FA and BP. Both methods reach the same nonzero ϵ_g, which is the minimal ϵ_g achievable as the student lacks the complexity to fully realize the rule.

We do observe that FA goes through two short plateau states before reaching this value, while BP does not. In Figure 16 the effect that an unrealizable scenario has on the order parameters R_{im} can be seen. Here, the blue lines represent the evolution of the order parameters R_{im} for BP and the red lines for FA. We can see that, for this particular initialization, BP breaks the symmetry faster and specializes. We observe that while both students phase out 2 teachers and specialize mostly to another teacher. They also maintain a positive overlap with one other teacher, corresponding with a minimum in the loss landscape. Figure 15a shows the inverse cosine alignment between $\boldsymbol{v}$ and $\boldsymbol{b}$. Interestingly, the alignment fluctuates at a low degree angle before settling at a final angle. The adjacent graph shows this alignment overlayed on top of the ϵ_g graph for FA from Figure 14a, showing again that the angle increases at loss plateaus. Of course, not many conclusions can be drawn from a single simulation, the graph in Figure 14b shows an average over $n = 10$ simulations with different random initial conditions for the feature vectors. Here we see that, on average, BP has a slight advantage as it seems that the plateau states are generally shorter.

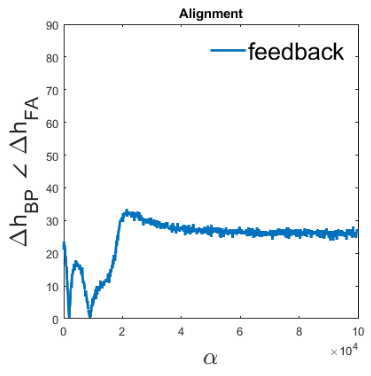


(a)

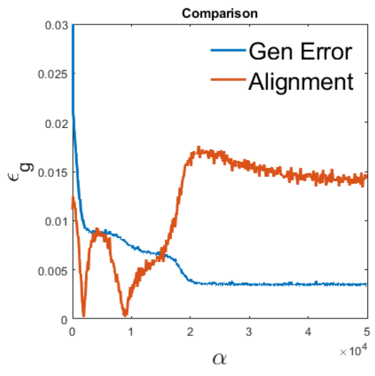


(b)

Figure 14: (a) Evolution of ϵ_g for a 500-4-1 sigmoidal teacher network and a 500-2-1 student network for a single simulation. Both methods have identical initial conditions. (b) Average ϵ_g after $n = 10$ simulations with different random initializations.



(a)



(b)

Figure 15: (a) Inverse cosine alignment between $\boldsymbol{v}$ and $\boldsymbol{b}$, representing the angle between the updates for FA and BP. For a 500-2-1 student network and a 500-4-1 teacher network. (b) The inverse cosine alignment overlayed on the evolution of ϵ_g for FA from Figure 14a
.

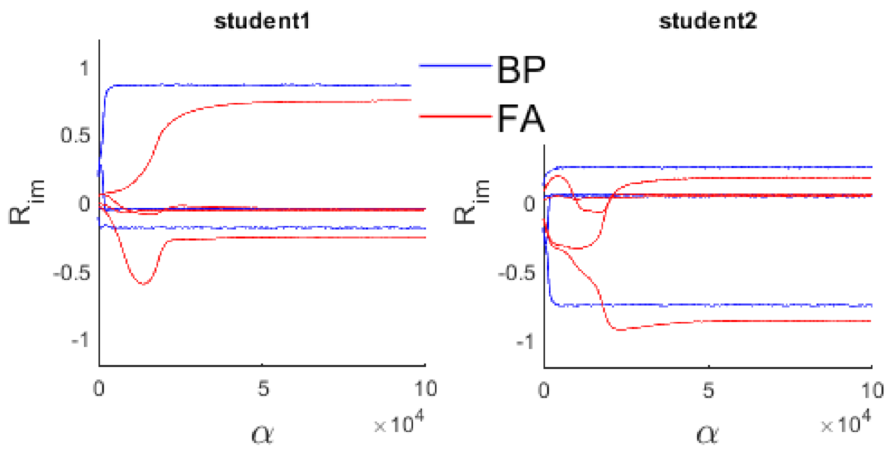
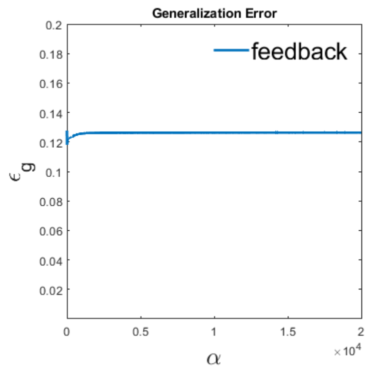


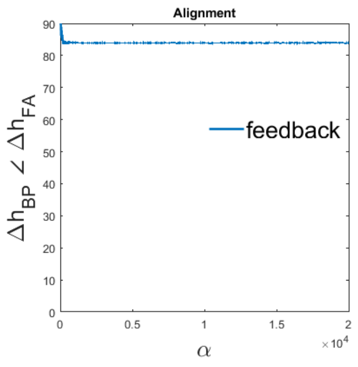
Figure 16: Order parameters R_{im} for BP and FA in an unrealizable scenarios with a 500-2-1 student network and a 500-4-1 teacher network, both with erf activation. The blue line represents BP, the red line represents FA.

3.2 ReLU Networks

When researching the performance of FA in ReLU networks, we have to take into consideration the signs of the random weights in $\boldsymbol{b}$. Because the random weights are fixed, when these signs do not correspond with those of the teacher weights in $\boldsymbol{v}^*$, the process cannot converge. For example, if a teacher weight in $\boldsymbol{v}^*$ is positive and a feedback weight in $\boldsymbol{b}$ is negative, the network will always drive the corresponding input-to-hidden weight vector in $\boldsymbol{J}$ in the opposite direction. For sigmoidal networks, this is not an issue due to the symmetry of the sigmoidal activation. However, for the non-symmetric ReLU activation, this becomes an issue. Using a simple example network with $K = M = 2$, we can see this non-convergence in Figure 17. Here, $\boldsymbol{v}^*$ for the teacher network contains a positive and a negative value, while the feedback vector $\boldsymbol{b}$ consists of two positive values. ϵ_g does not converge as FA sends one of the $\boldsymbol{J}_i$ in the wrong direction. The alignment plot next to it shows that the alignment between the forward weights and the feedback vector stays low. The angle $\Delta\mathbf{h}_{FA}\measuredangle\Delta\mathbf{h}_{BP}$ between the two vectors stays close to 90°, as the network can not learn how to use the feedback vector effectively. If BP is used in the update steps, it does not matter if the signs align between $\boldsymbol{v}$ and $\boldsymbol{v}^*$ on initialization, over the course of training, they can flip and learning can converge.



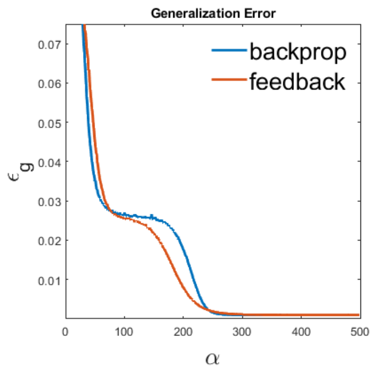
(a)



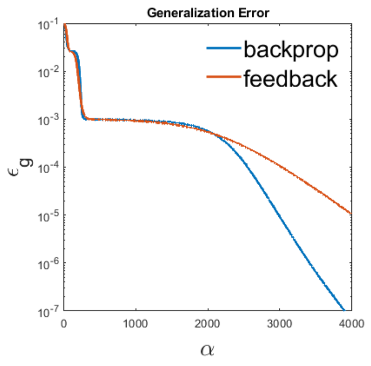
(b)

Figure 17: (a) Evolution of ϵ_g for a ReLU network with an unusable feedback vector, where the signs of the vector components in $\boldsymbol{b}$ do not align with the signs of the components in $\boldsymbol{v}^*$. (b) Inverse cosine alignment between forward weights $\boldsymbol{v}$ and $\boldsymbol{b}$

Because of this problem, to use FA in combination with a ReLU network, one either has to know something about the rule, or an overrealizable scenario has to be considered. The first option is rather unlikely in realistic machine learning situations. Possibly from a Bayesian point of view, we could say we have some prior knowledge of the weights. If all the weights of $\boldsymbol{b}$ and $\boldsymbol{v}^*$ are set to positive values, the issue with the signs would disappear. For the following experiment, we do exactly this, implying we have some prior knowledge about the signs of the hidden-to-output weights of the rule network. The graphs in Figure 18 show the evolution of ϵ_g with this prior knowledge about the rule with $\eta = 0.05$. From the graph on the left with a linear scale, we observe that, unlike in the previous example with misaligned signs, FA shows convergent behaviour. The generalization error goes to zero after a short plateau state in around the same number of discrete timesteps α as BP. The graph beside it shows the same result on a logarithmic scale.



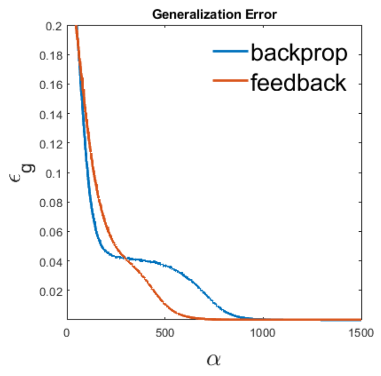
(a)



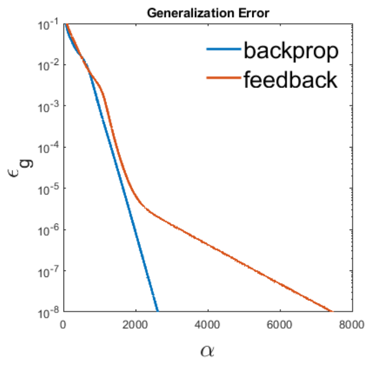
(b)

Figure 18: (a) Evolution of ϵ_g for a 500-4-1 ReLU network with prior knowledge about the signs of the teacher weights, (b) The same graph on a single logarithmic scale.

Aside from the output, the rule is generally unknown. One could run different initializations in the hopes that for one of them, the process will converge. This is highly unlikely though, especially in networks with a large number of adaptable weights. The most realistic option is to use an overrealizable scenario, where $K > M$. In such a situation, there is a higher chance that enough of the signs in $\boldsymbol{b}$ are correspond with those of the teacher weights $\boldsymbol{v}^*$. To ensure a 100% chance of this correspondence, we need $K \geq 2M$, the width of the hidden layer in the student has to be at least twice the size of that of the teacher. If we then initialize $\boldsymbol{b}$ with an equal amount of positive and negative components, at least M of these signs will align. The students corresponding with these equalities can then specialize to a teacher. The other student weights can then either be seen as redundant by the network, or students might share specialization if multiple signs correspond. For this experiment, we use a 500-4-1 student network and a 500-2-1 teacher network. Here $K > M$, the student has more complexity than the teacher. In Figure 19a we see the evolution of the generalization error ϵ_g for both BP and FA. This graph represents a single simulation with identical initial conditions for both methods, using $\eta = 0.05$. We observe that the usage of FA results in an earlier escape from the loss plateau compared to when BP is used. In the graph in Figure 20a, the inverse cosine alignment representing the updates made by BP and those made by FA is shown and in the neighbouring graph, this angle is overlayed on top of the ϵ_g graph from Figure 19a. After an initial decrease, the angle increases again when the loss starts to slow down. Following is a short semi-plateau, when this plateau is broken, the angle decreases again and settles at a final value when ϵ_g is asymptotically reduced to zero. Figure 21 displays the evolution of the order parameters R_{im} for BP in blue and for FA in red. Again, both methods behave almost exactly alike. From the graph, it looks as though both student 2 and 4 share specialization with the other students. However, two of the four hidden-to-output weights are zero at the end of training, two student units are phased out. Even if there is the expected specialization, the order parameters R_{im} do not necessarily have to settle at -1 or 1. This is due to the linear nature of the positive part of the ReLU function and the linear output unit. There are different combinations possible between $\boldsymbol{J}_i$ and v_i that still converge to a correct solution. The graph in Figure 19b shows the generalization error on a logarithmic scale. These plots are averaged over $n = 10$ simulations with random initializations. Here, we can see that FA and BP perform similarly until a certain point in the learning process. At this point, the decrease of ϵ_g for FA goes from fast exponential decrease to a slower exponential decrease.

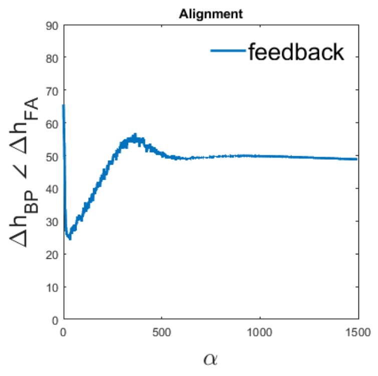


(a)

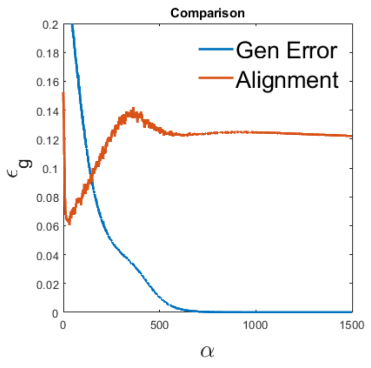


(b)

Figure 19: (a) Evolution of ϵ_g for both FA and BP for a 500-2-1 teacher network and a 500-4-1 student network with ReLU activation. Single simulation with identical initial conditions for both methods. (b) Generalization error for BP and FA on a logarithmic scale, averaged over $n = 10$ simulations with random initial conditions.



(a)



(b)

Figure 20: (a) Inverse cosine alignment for the updates made by BP and those made by FA for a 500-2-1 teacher network and a 500-4-1 student network with ReLU activation. (b) The alignment overlayed on the evolution of ϵ_g for FA from Figure 19a

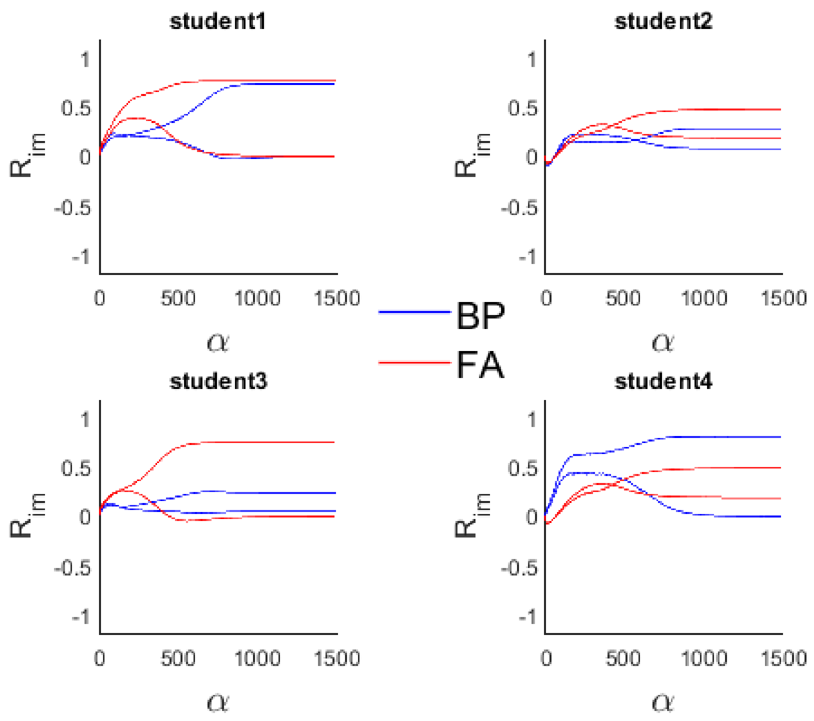


Figure 21: Order parameters R_{im} for a 500-2-1 teacher network and a 500-4-1 student network, both with ReLU activation. Blue represents BP, red represents FA.

3.3 Discussion

From the experiments described in this section, it is shown that FA works even for shallow networks with one hidden layer and a small number of adaptable hidden-to-output weights. We observed that, in general, performance is very similar for differently realizable scenarios. This is indicated by the averages in Figure 5b, 10b and 14b. In some situations, the use of FA might be useful in breaking unspecialized plateaus states earlier. However, this is not necessarily always the case. At times, the opposite situation may even occur, where using FA results in a longer plateau state compared to using BP. This is also visible in Figure 5b showing an average over $n = 10$ simulations. For at least one of these simulations, training using FA goes through a long plateau state before ϵ_g decreases again rapidly and the learning process converges. In overrealizable scenarios, there may be a more systematic correspondence between using FA and earlier escapes from plateau states. However, in underrealizable scenarios, the opposite may be true. As shown in Figure 14a, where FA goes through two plateau states before reaching a minimum. There is some slight oscillation at this value of ϵ_g. Because the rule is unrealizable and we use a fixed learning rate with on-line gradient descent, the value of ϵ_g will always oscillate around this minimum. Figure 6b for a perfectly realizable scenario and Figure 12b for an overrealizable scenario show what happens to the alignment between the updates made by BP and those made by FA in plateau states where the decrease in loss can come to a complete halt. At each point in training when a plateau is reached, the angle $\Delta\mathbf{h}_{FA}\measuredangle\Delta\mathbf{h}_{BP}$ increases, i.e. the updates made by BP de-align with the updates made by FA. When a plateau is broken, this angle swiftly decreases again. Only after all plateaus are broken and ϵ_g is asymptotically decreasing to 0, does the angle settle at a final value. In an unrealizable scenario, the situation is similar. However, the angle can decrease to even lower values before increasing again when a loss plateau occurs, as seen in Figure 15b. Here, the angle even goes down to 0, meaning that $\boldsymbol{v}$ and $\boldsymbol{b}$ are completely aligned, both are pointing in the identical direction. As mentioned previously, the alignment of the updates made using BP with those of FA changes depending on the different scenarios shown in section 3. In most situations, the initial decrease is large at the beginning of learning before sharply decreasing in the first few timesteps α. After this, some fluctuation occurs, where every increase of the angle represents the system being stuck in a fixed point in the learning dynamics. In perfectly realizable scenarios, the final angle settles after some initial fluctuation. In unrealizable learning situations, the evolution of the angle shows that it reaches very small values, before increasing again and settling at a higher angle. In overrealizable scenarios, there is some initial fluctuation again. After this short oscillation, the angle starts slowly increasing until a final non-changing angle is reached. This slow increase represents some components of the $\boldsymbol{v}$ being set to zero, which either increases or decreases the alignment between those components of $\boldsymbol{b}$ and $\boldsymbol{v}$, how these weights were aligned beforehand. The final angle is generally higher than its realizable and unrealizable counterparts. This is logical, as the number of redundant parameters in $\boldsymbol{v}$ increases, the corresponding components in $\boldsymbol{b}$ have no alignment with these parameters.

In overrealizable scenarios, a recurring phenomenon is the change in the exponential decrease of ϵ_g to a slower, but still exponential decrease. We compared the results for an overrealizable shallow sigmoidal network to that of Frederieke Richert et al. [15] where they discover a different rate of convergence. This convergence was proportional to a power-law decrease, where $\epsilon_g \propto \frac{1}{\alpha^2}$. This is not confirmed by these experiments. The main and most important difference between their experiments and the ones shown in this thesis, is that they use a particular normalization of the student output which allows all students to learn and replicate one of the teachers. Therefore, this rate of convergence in overrealizable shallow networks only happens when multiple students specialize to one teacher. In our scenario, there is no normalization factor and reducing components from $\boldsymbol{v}$ to zero becomes the preferred solution.

When a different variant of the gradient for FA in shallow networks with a sigmoidal output unit was used, the process still converged to zero ϵ_g. By changing the inner product of the derivative of the activation function, the magnitude of the updates changes as well, as the result of $\boldsymbol{g}(\cdot)$ is different. Using this variation, the alignment between the updates made by BP and those made by FA barely changes, visible in Figure 9. This is likely why the performance is very similar compared to the normal variation of FA, as is shown in Figure 8.

For ReLU networks, the issue with the signs of $\boldsymbol{v}^*$ and $\boldsymbol{b}$ allowed us to study three situations. When these signs do not align, the process will not converge as some components of $\boldsymbol{b}$ are unusable. One might compare this to the unrealizable scenario from the experiments using erf networks. The difference being that all components of $\boldsymbol{b}$ in unrealizable erf networks are still usable in some manner. We may have prior knowledge of the signs of the hidden-to-output teacher weights $\boldsymbol{v}^*$. In this situation, the performance of both methods is very similar, as seen in Figure 18. The entire learning process becomes rather trivial, as both FA and BP do not need large α to reach very low ϵ_g. This rather unrealistic experiment at least indicates that the usage of FA in the update steps does work with ReLU networks. The other, more realistic scenario is an overrealizable one, where $K \geq 2M$. Any redundant students will be set to zero and ϵ_g will converge to 0 at similar rates. The graph in Figure 19b shows that BP and FA converge to 0 ϵ_g at a similar exponential decrease. For FA, this exponential decrease slows down at some point. This slowdown only happens for 2 out of the $n = 10$ simulations, it is unclear as to why this occurs. However, it seems somewhat insignificant, as $\epsilon_g < 10^{-5}$ at that point in the training process.

4 Results for Deep Networks

In this section, a number of experiments are shown concerning the training of 3-layer networks. These networks have two hidden layers, with the synaptic hidden-to-hidden weight matrix $\boldsymbol{W}$ connecting the two. This means that if we use FA, apart from $\boldsymbol{v}$ being replaced by $\boldsymbol{b}$ in the backpropagation step, we also replace $\boldsymbol{W^T}$ by $\boldsymbol{B}$. The teacher weights $\boldsymbol{W}^*$ and the forward weights $\boldsymbol{W}$ are sampled from a normal distribution $\mathcal{N}(0, 1)$, as are the random feedback weights $\boldsymbol{B}$. The student-teacher overlaps are initialized as explained above for shallow networks. Again, we supply the system with a stream of examples with i.i.d. components from $\mathcal{N}(0, 1)$, using stochastic on-line gradient descent to perform the weight updates.

4.1 Sigmoidal Networks

For the first experiment with a deeper structure, we go back to a neural network with erf activation in the hidden units. We reduce the input dimension from $N = 500$ to $N = 250$, to speed up convergence. Moreover, because the network has more layers, it is more complex, the input dimension can be reduced while still observing phenomena like long plateau states. Again, the first scenario to be studied is a perfectly realizable one. In section 2, the structure of the deep networks used in the following experiments is shown. For a student network, the first hidden layer will have K hidden units, the second layer has $K/2$ hidden units with a single linear output unit. A teacher network has an identical structure, with M and $M/2$ units in the first and second hidden layer, respectively. Here, $K = M = 4$, both student and teacher are 250-4-2-1 erf networks. In the original FA paper by Lillicrap et al. [8], it is mentioned than when the forward weights are initialized at or near zero, it promotes faster convergence to a correct local minimum. The feedback matrix $\boldsymbol{B}$ begins to act like a local pseudo-inverse of $\boldsymbol{W}$. This does not mean that for different initializations of $\boldsymbol{W}$ and $\boldsymbol{v}$, the process would not converge. However, as this promotes faster convergence, forward weights are initialized $\boldsymbol{W}$ and $\boldsymbol{v}$ to 0 when FA is used. Because this is a more complex network with an increased number of adaptable weights, the learning rate is reduced to $\eta = 0.02$. There exists no analytical solution for the average prediction error ϵ_g on new input data when 3-layer networks are considered, therefore we calculate it empirically as an average over N_s random test inputs, as shown in Equation (33). In Figure 22 we can see the generalization error on a linear scale for a single simulation with BP and FA. The initial conditions are identical for both methods. As ϵ_g is calculated using an empirical average, the graph lacks smoothness compared to the analytical solution for ϵ_g used in section 3. However, it is clear that both FA and BP observe a plateau state at a similar ϵ_g of differing lengths. For this particular initialization, the system breaks out of this unspecialized state faster when FA is used. The graph in Figure 24 shows the order parameters R_{im} for BP in blue and FA in red. The long symmetric state corresponding to the loss plateau of ϵ_g is clearly visible in student 1. In Figure 23, the inverse cosine alignment between the updates made by BP and those made by FA is shown. After a sharp decrease, there is some initial fluctuation at the beginning of the learning process, followed by an optimal final angle for which the system reaches convergence. The adjacent graph shows the inverse cosine alignment overlayed on the ϵ_g graph of Figure 22 for FA. The initial increase of the alignment happens exactly at a plateau state, as does the second increase.

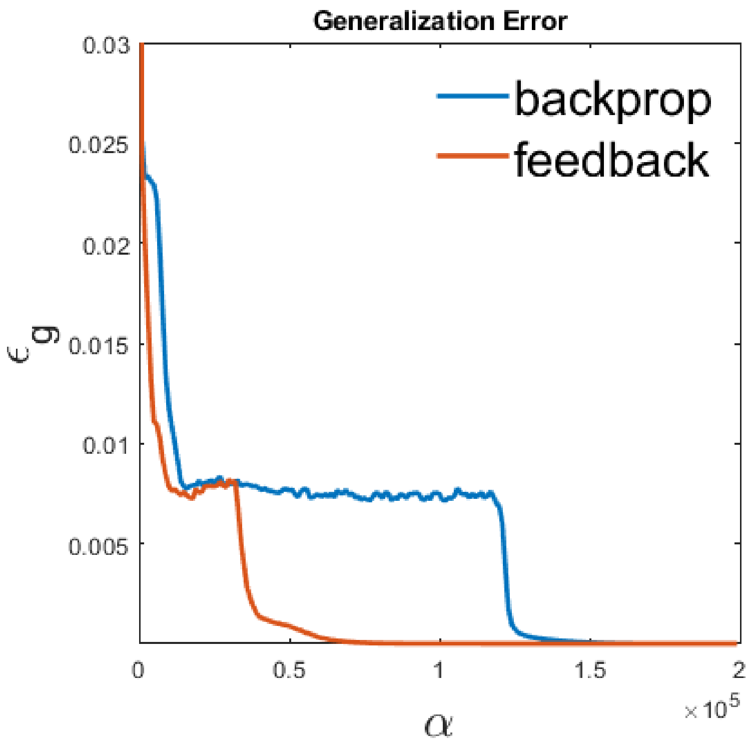
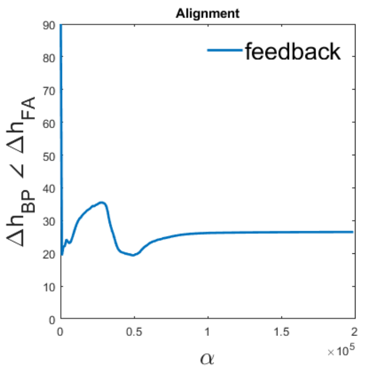
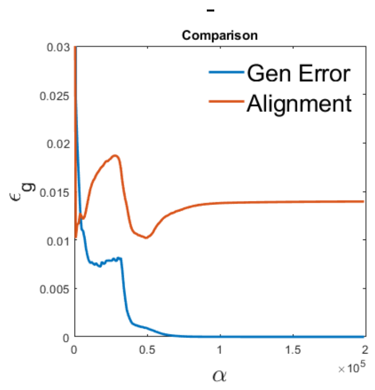


Figure 22: Evolution of ϵ_g for a 250-4-2-1 student and teacher network with erf activation. The results are for single simulation with identical initial conditions for FA and BP.



(a)



(b)

Figure 23: (a) Inverse cosine alignment angle between the updates made by BP and those made by FA for a 250-4-2-1 student and teacher network with erf activation. (b) The inverse cosine alignment overlayed on the evolution of ϵ_g for FA.

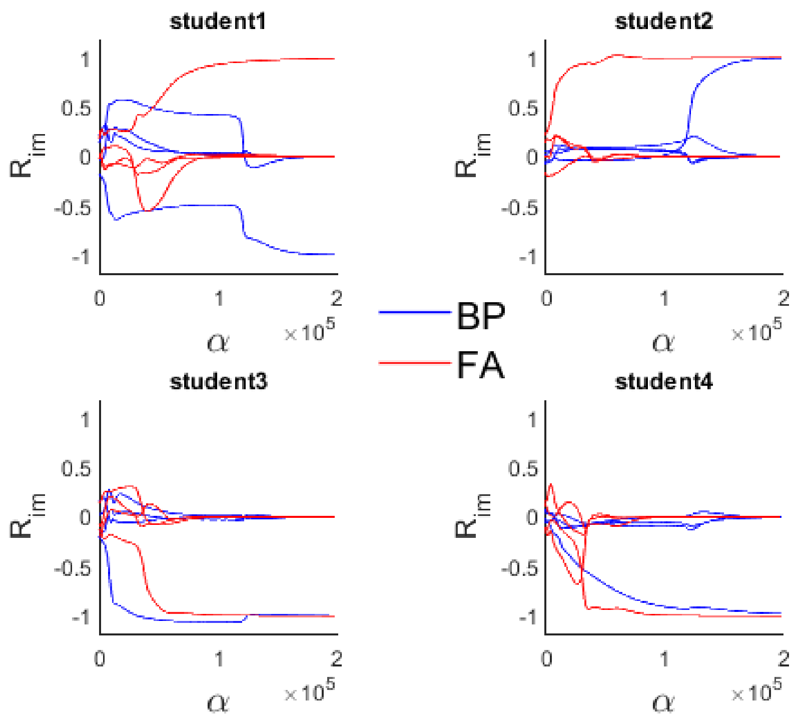
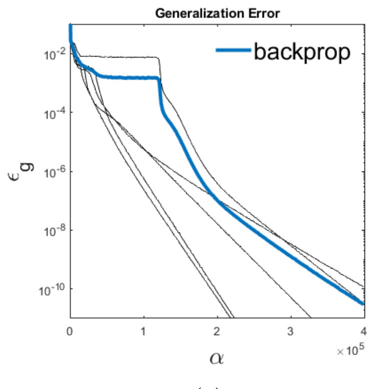
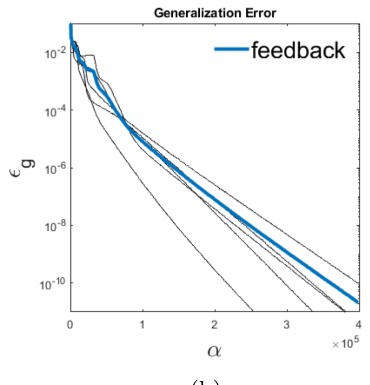


Figure 24: Evolution of the order parameters R_{im} in a 250-4-2-1 student and teacher network. BP is shown in blue, FA in red.

To get an idea of the overall performance of FA in realizable networks with 2 hidden layers, α is increased and an average is taken over $n = 5$ simulations. This can be seen in Figure 25. Shown is a side-by-side view of ϵ_g for both methods on a single logarithmic scale. Individual simulations are displayed in grey and the blue line represents the average over all $n = 5$ simulations. The average performance of both methods is quite similar, with some plateaus of differing length in the early stages of learning, followed by a quick exponential decrease to zero error.



(a)



(b)

Figure 25: (a) Evolution of ϵ_g for BP in a 250-4-2-1 network. The results are averaged over $n = 5$ simulations, the individual simulations can be seen in grey. (b) The same situation for FA.

In the gradient from equation (16), $x_2 = \boldsymbol{W}\boldsymbol{g}(\boldsymbol{J}\boldsymbol{\xi}^\mu)$. As for the gradient for the shallow network with a non-linear output unit, this gradient requires individual nodes in the first hidden layer to know the incoming weights of the other nodes in that layer. This means that when a node i in the second hidden layer is updated, it needs to have knowledge of all incoming weights W_j. We could argue once more, that this is biologically implausible and replace $\boldsymbol{W}$ by $\boldsymbol{B}^T$ in the derivative of the activation function. The resulting partial gradient will be

$$\frac{\partial}{\partial \boldsymbol{W}} e^\mu = \delta^\mu \boldsymbol{b} \odot \boldsymbol{h}'(\boldsymbol{B}^T \boldsymbol{g}(\boldsymbol{J}\boldsymbol{\xi}^\mu)). \tag{38}$$

With the same setup as described above, we run another experiment using this variant of the gradient. In Figure 26, we see the evolution of the ϵ_g for three simulations with identical initial conditions and different learning rates. The graph shows an entirely different situation compared to the results for the shallow variant. After some initial decrease in ϵ_g, the learning process diverges and the generalization error increases again rapidly before fluctuating at a high value.

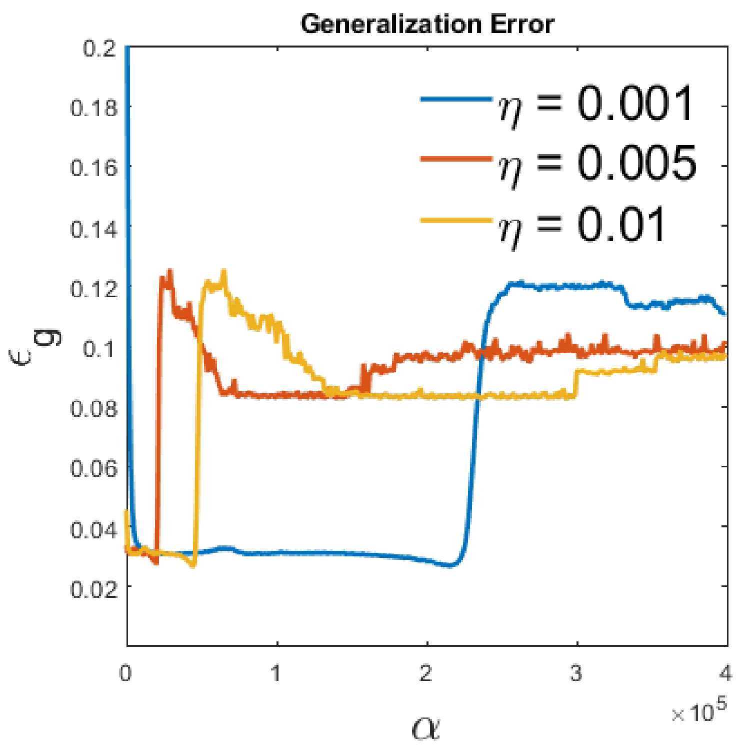


Figure 26: Evolution of ϵ_g for a 250-4-2-1 student and teacher erf network using a different variant of the gradient, where the input for that activation derivative is changed. The graph shows three individual simulations with identical initial conditions and different learning rates.

4.1.1 Overrealizable, $K > M$

Again, we are interested in an overrealizable scenario, where the student network has increased complexity compared to the teacher network, $K = 8$ and $M = 4$. The 250-4-2-1 teacher network is identical to the one used in the previous experiment. However, the 250-8-4-1 student network has twice the complexity regarding the number of adaptable weights. Again, we use $\eta = 0.02$. After learning, generally only 4 students remain that have optimal overlap with the 4 teachers. The other 4 students are likely phased out and set to zero. Because of the large number of teacher weights $\boldsymbol{C}$, we refrain from showing the evolution of the order parameters R_{im} here. Figure 27 shows the progression of ϵ_g on a linear scale for both FA and BP. Again, for a fair comparison, both methods use identical initial conditions. Here, both FA and BP reach a plateau state at a similar value of ϵ_g. For these initial conditions, FA breaks the symmetry in fewer discrete timesteps than its counterpart. In Figure 28a the angle over time between the updates made by BP and those made by FA is shown. Next to this graph, this alignment is overlayed on top of the ϵ_g graph for FA from the previous figure. Again, we see that when the short loss plateau is reached at the early learning stage, the angle will increase. After this plateau, the angle slightly decreases again before slightly increasing again to the angle corresponding with an exponential convergence rate.

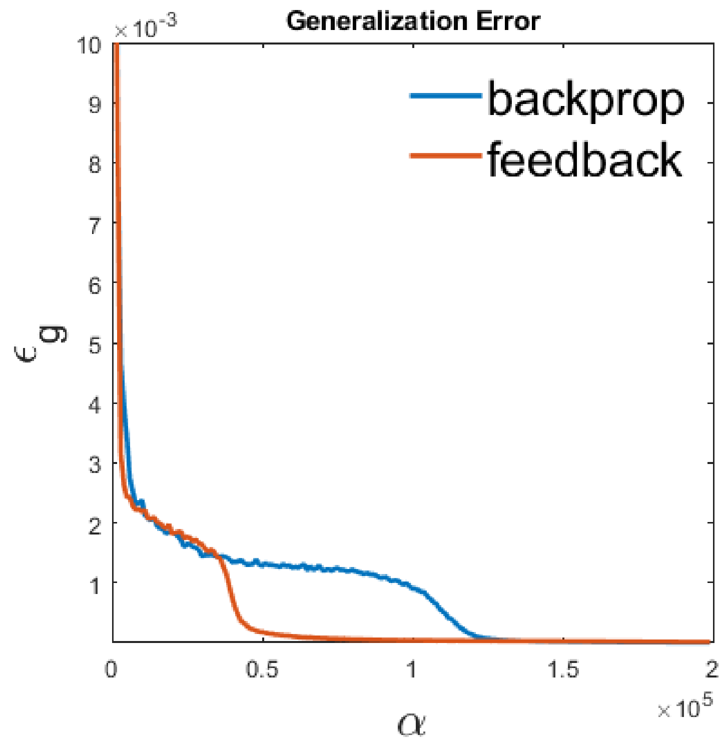


Figure 27: Evolution of ϵ_g for a 250-4-2-1 teacher network and a 250-8-4-1 student network with erf activation. The results are for a single simulation, both BP and FA have identical initial conditions

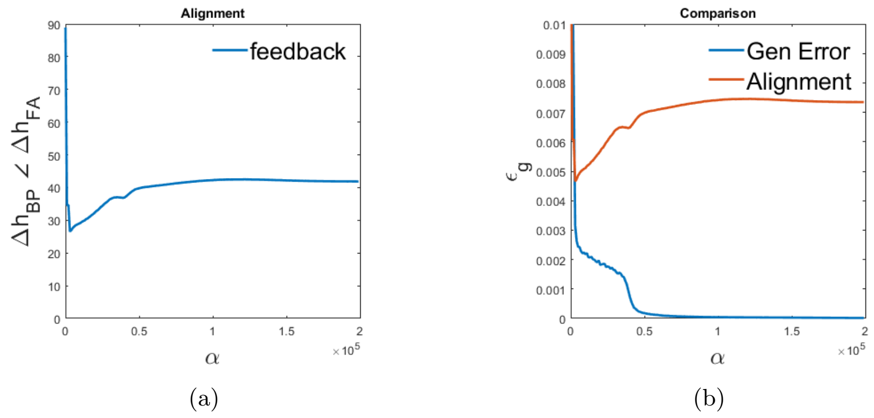
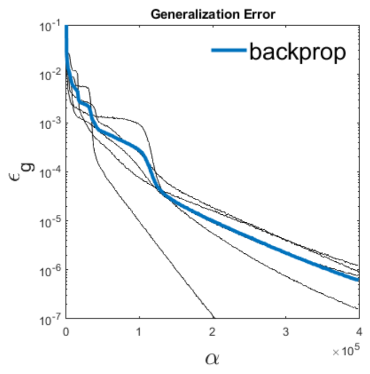
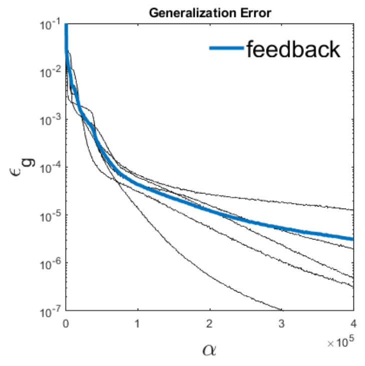


Figure 28: (a) Inverse cosine alignment angle between the updates made by FA and those made by BP, for a 250-8-4-1 student network and a 250-4-2-1 teacher network with erf activation. (b) The inverse cosine alignment overlayed on the evolution of ϵ_g for FA from Figure 31a.

In Figure 29 we see the results for both FA and BP with ϵ_g on a logarithmic scale. Convergence rates are similar, with BP having a slight advantage. However, the individual simulations of FA go through shorter and fewer plateau states. We want to take a closer look at the decrease of ϵ_g for two individual simulations for both BP and FA. More specifically, two simulations which seem to behave asymptotically in the single logarithmic plot. Figure 30 shows a comparison of the single and double logarithmic plot. In the single log-plot, after breaking of the unspecialized states, the decrease of ϵ_g seems to be asymptotically bound. As α increases, the decrease of ϵ_g is no longer exponential. The tail of the double-log plot appears to be straight, which would indicate that there is power-law convergence $\epsilon_g \propto 1/\alpha^2$, this is confirmed by measuring the slope. For these two individual simulations, multiple students are sharing specialization to a teacher. This corresponds with the findings in [15], where they observe a power-law rate of convergence in situations with shared specialization.

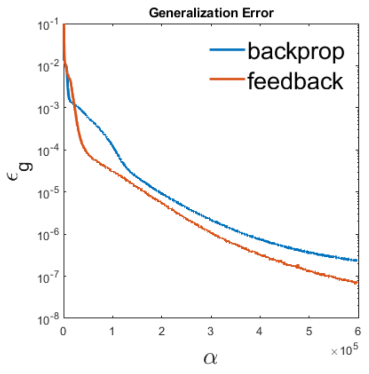


(a)

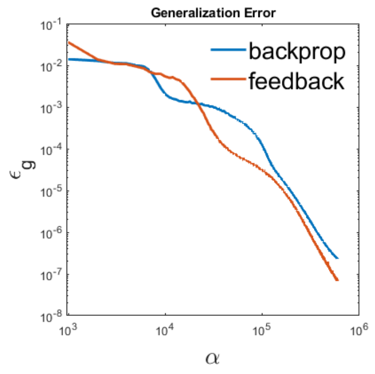


(b)

Figure 29: (a) Evolution of ϵ_g for FA in a 250-8-4-1 student network with a 250-4-2-1 teacher network, both with erf activation in the hidden units. The results are averaged over $n = 5$ simulations, the individual simulations can be seen in grey. (b) The same situation for BP.



(a)



(b)

Figure 30: (a) Two individual simulations for BP and FA taken from Figure 29 where the decrease of ϵ_g is asymptotically bound on a single logarithmic scale. (b) Double logarithmic plot of the same individual simulations, the tail of both graphs follow a straight line, indicating that $\epsilon_g \propto \frac{1}{\alpha^2}$.

4.1.2 Unrealizable, $K < M$

In the analysis of an unrealizable scenario for a deep network, the previous situation is flipped. Here, $N = 250$, $K = 4$ and $M = 8$, the 250-4-2-1 student network lacks complexity in comparison to the 250-8-4-1 teacher network. The same learning rate is used as in the overrealizable and perfectly realizable scenario, $\eta = 0.02$. As with shallow networks, it is expected a student will specialize to multiple teachers as the rule cannot be perfectly realized. Figure 31a displays the evolution of ϵ_g for a single run with identical initial conditions for FA and BP. Both methods go through a short plateau state before reaching the same minimal ϵ_g at which the students have specialized optimally to the teachers. When FA is used, this value of ϵ_g is reached at a slightly later stage, due to the unspecialized state being slightly longer. In Figure 32a we can see the corresponding alignment between the updates made by BP and those made by FA. Again, as in the alignment graph for a shallow unrealizable situation, the alignment dips to some low degree angle before increasing again and settling at a final angle. The neighbouring graph shows this angle of alignment overlayed on top of the ϵ_g graph for FA from Figure 31a. Again, in the short plateau state, the alignment increases before decreasing swiftly again when the loss continues decreasing. The graph in Figure 31b shows an average over $n = 5$ simulations for both methods, the performance of both methods looks quite similar, with BP having a slight advantage.

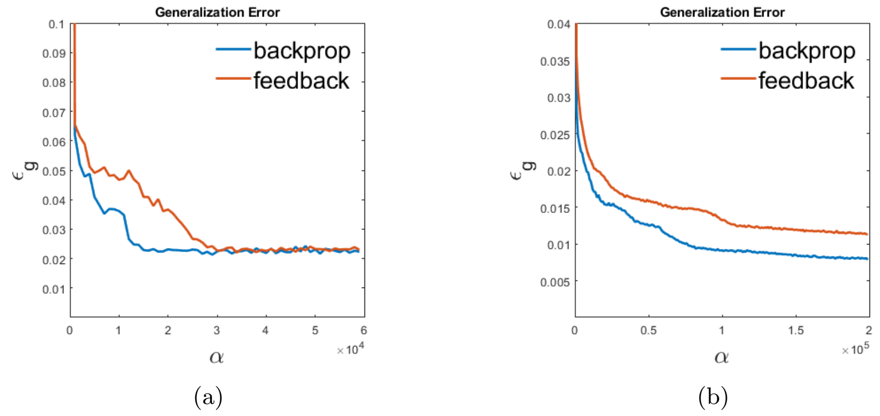
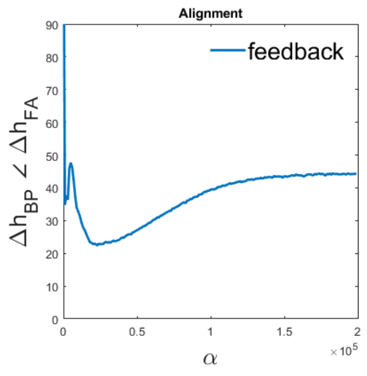
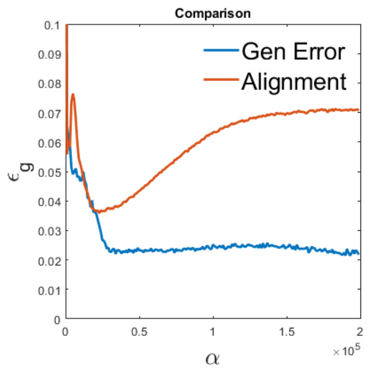


Figure 31: (a) Evolution of ϵ_g using a 250-4-2-1 student network and a 250-8-4-1 teacher network with erf activation. Resulting graphs are for a single run with identical initial conditions for BP and FA. (b) Similar graph for an average of $n = 5$ simulations with different random initial conditions for both methods.



(a)



(b)

Figure 32: (a) Inverse cosine alignment angle between the updates made by FA and those made by BP, for a 250-4-2-1 student network and a 250-8-4-1 teacher network with erf activation. (b) The inverse cosine alignment overlayed on the evolution of ϵ_g for FA from Figure 31a
.

4.2 ReLU Networks

We are also interested in the effectiveness of FA in deeper networks with ReLU activation in the hidden units. However, we run into the same issue as before when the signs of $\boldsymbol{b}$ and $\boldsymbol{B}$ do not correspond with those of the teacher weights $\boldsymbol{v}^*$ and $\boldsymbol{W}^*$. If we were to run a scenario with $K = M$, it is highly likely that the signs of the feedback weights do not align with those of the teacher weights. The resulting ϵ_g graph would look similar the graphs shown in Figure 17. We could assume some prior knowledge of the teacher weights again and initialize both the feedback weights $\boldsymbol{b}$ and $\boldsymbol{B}$, and the teacher weights $\boldsymbol{v}^*$ and $\boldsymbol{W}^*$ with only positive weights. If we do this, the rule becomes very trivial to realize, as is shown in Figure 33, showing the evolution of ϵ_g on a linear and logarithmic scale. Even with a relatively small $\eta = 0.01$, both methods go through one short plateau state before reaching convergent behavior in a small number of discrete timesteps α.

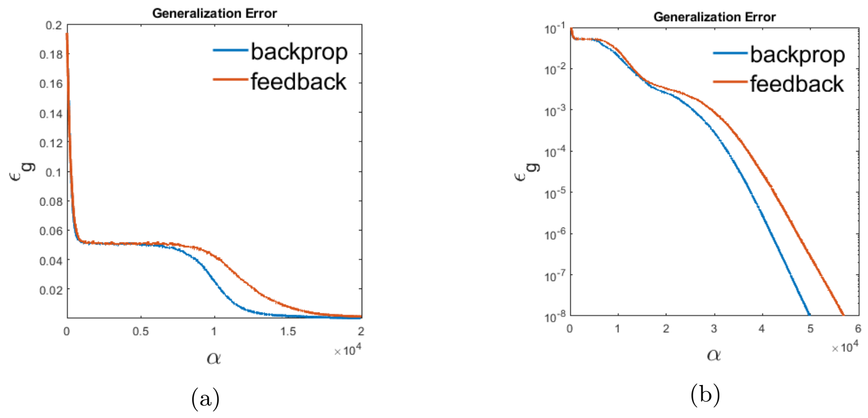


Figure 33: (a) Linear plot of the evolution of ϵ_g for a 250-4-2-1 ReLU network with prior knowledge of the teacher weights, (b) Logarithmic plot of the same situation.

We are more interested in the dynamics of an overrealizable scenario with an overly complex student, as this is more realistic than the student having prior knowledge of the rule it is trying to learn. For erf networks, we had the option of initializing the forward weights to zero. This not possible for ReLU, as the derivative at $\text{ReLU}'(0)$ is manually defined as 0. Therefore, the derivative of the activations in the gradient calculation will always be zero when the forward weights are initialized as such and no updates will be made to the weights. For the following experiment, we use the same input dimension $N = 250$, as we used for sigmoidal networks. However, the complexity of the student network is increased even further. Using a 250-4-2-1 teacher network and a 250-10-5-1 student network, $K = 10$ and $M = 4$. A smaller learning rate is used, $\eta = 0.01$. Some preliminary testing indicated that the student overlaps Q_{ik} need not be initialized to high values to observe long symmetric states. Therefore, the student-student overlaps are changed to $Q_{ik} = 0$ for $i \neq k$. In Figure 34 we see a comparison of the evolution of ϵ_g for both BP and FA. This graph represents a single run for both methods with identical initial conditions. Here, BP goes through multiple plateau states before the error asymptotically reduces to zero. When FA is used, it suffers from only one short plateau state before the same convergent behaviour is shown. Figure 35a displays the alignment angle for updates made by BP and those made by FA. Immediately noticeable is the final angle being considerably higher than the same angle for overrealizable deep erf networks. We can get a more detailed comparison by increasing α and showing the evolution of ϵ_g on a logarithmic scale, this is shown in Figure 36. The average of $n = 5$ simulations is shown in blue, the individual simulations are included in grey. For many runs, using BP can result in long unspecialized states. When FA is used, the plateau states are fewer and significantly shorter.

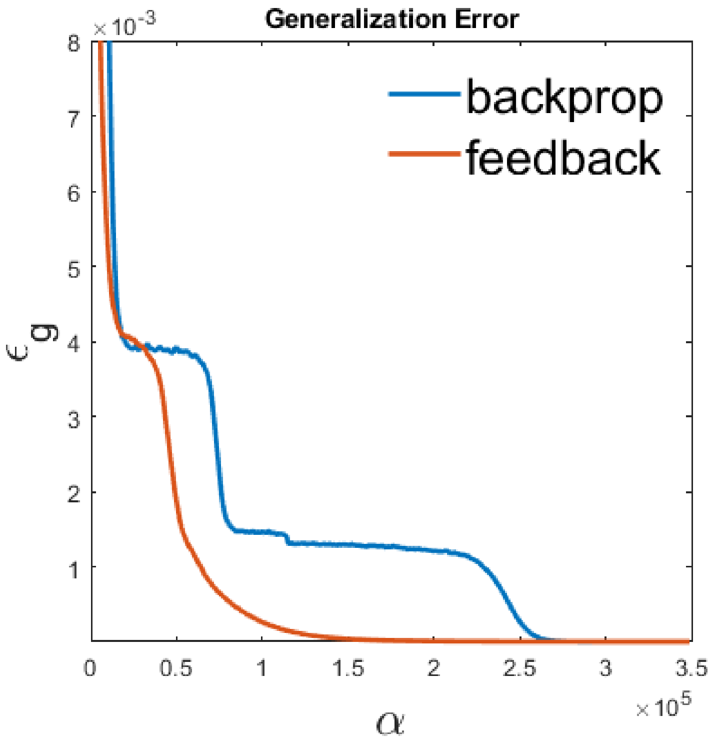


Figure 34: Generalization error ϵ_g for a single run with identical initialization, an over-realizable scenario with a 250-4-2-1 teacher network and a 250-10-5-1 student network with ReLU activation.

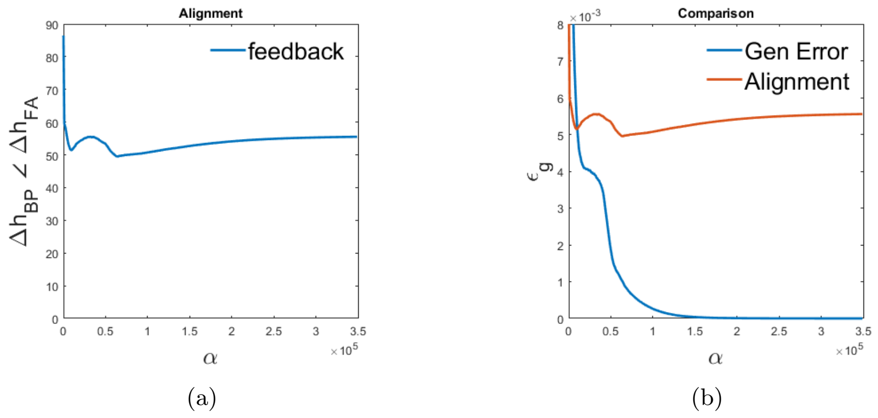
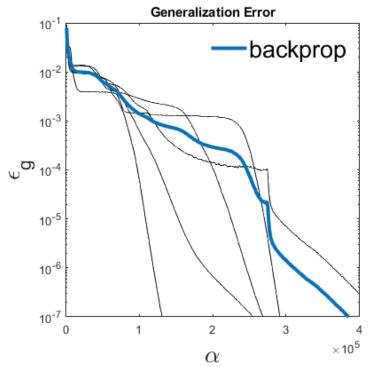
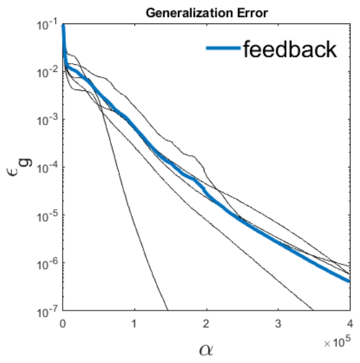


Figure 35: Alignment angle between the updates made by BP and by FA for an over-realizable scenario with a 250-4-2-1 teacher network and a 250-10-5-1 student network with ReLU activation.



(a)



(b)

Figure 36: (a) Empirical generalization error ϵ_g for BP for a 250-4-2-1 teacher network and a 250-10-4-1 student network with ReLU activation. The results are averaged over $n = 5$ simulations, the individual runs can be seen in grey. (b) The same situation for FA.

4.3 Discussion

For deeper erf networks with 2 hidden layers, the performance of FA is comparable to BP in many situations. In perfectly realizable and overrealizable scenarios shown in this section, FA can be at least as effective as BP. Figure 25 confirms this for realizable scenarios, as the average decrease of ϵ_g is very similar once all plateau states are broken. For particular initial conditions, we can see that FA can be used to get out of a loss plateau earlier. This fact is showcased in Figure 22 showing the rapid decrease of ϵ_g after the system breaks the symmetry. This rapid decrease comes earlier in the learning process for FA. We can see exactly what happens in Figure 24, showing the order parameters R_{im} in blue for BP. In the early stages of learning, student 1 tries to specialize and anti-specialize to 2 teachers simultaneously, whereas student 2 has no specialization. For FA, shown in red, the symmetry in R_{im} is resolved at an earlier stage. As in the results for shallow networks, when a plateau state is reached, the updates made by FA de-aligns with the updates made by BP, shown in Figure 23b, 28b and32b. In overrealizable deep erf networks, FA breaks the plateau states earlier on many individual simulations in which BP does not. This can be seen in Figure 29, where the individual runs in grey of FA reach a state of convergence at an earlier stage of learning than most of the individual runs for BP. However, the average in blue shows that BP has a faster decrease in ϵ_g. This could be due to some fortunate initializations for BP. Moreover, this faster decrease seems rather insignificant, as $\epsilon_g < 5^-5$ at the point in the training process where the loss decrease of BP dips below FA. After all symmetric states are broken, we can see the decrease of ϵ_g going from an exponential decrease to a smaller exponential decrease. For some initial conditions, using both FA and BP, the decrease of ϵ_g on a single logarithmic scale can be asymptotically bound. In Figure 30a, two individual runs which have this property are shown. This indicates that the decrease of ϵ_g has stopped following an exponential decrease, instead having an algebraic power-law decrease $\alpha \propto \frac{1}{\alpha^2}$. The graph in Figure 30b confirms this suspicion, by showing the evolution of ϵ_g on a double logarithmic scale for these two simulations. The tail of both graphs follow a somewhat straight line, indicating that the rate of decrease changed. The measured slope confirms that ϵ_g has stopped decreasing exponentially. This situation solely occurs when multiple students share specialization with a teacher. This corresponds with the experimental results for Soft Committee Machines performed by Frederieke Richert et al.[15], showing that this situation can also occur in deeper networks.

If the different variant of FA is used where the gradient for the activation in the second layer is changed, it does not show convergent behaviour. Instead, the learning will diverge after some point and ϵ_g stays high, as showcased in Figure 26. Results for various learning rates are shown to indicate that the divergence is not related to a poorly chosen η. At the point where ϵ_g starts rapidly increasing, the absolute values of the weights in $\boldsymbol{W}$ start increasing rapidly as well. This divergent behavior is rather confusing, as changing the weights in $\boldsymbol{g}'(\cdot)$ only changes the magnitude of the updates. It might indicate that the $\boldsymbol{g}'(\cdot)$ term conveys important information that contributes to the alignment for deeper networks. This information apparently gets altered in a manner that does not allow the update steps to represent backpropagation. Another possible reason lies in the fact that in the update for the second layer weights, both $\boldsymbol{B}$ and $\boldsymbol{b}$ are used. This may hinder the ability for the system to find the proper alignment with the feedback weights.

In the overrealizable scenarios using networks with ReLU activation, we again observe this faster breaking of plateau states when FA is used. These experiments shown in Figure 34 and 36, show BP suffering from more plateaus states of increased length compared to the results for FA. However, it is unclear if this is due to the fact that the situation is overrealizable, or simply because K is set to a high number. Resulting in more student vectors that need to learn the rule. The convergence following the plateau states is faster for BP. However, when the ϵ_g is already sufficiently low, the faster decrease of ϵ_g seems less important compared to the loss decrease slowing down significantly in the early stages of learning. Interestingly, the angle representing the alignment between updates made by BP and those made by FA in Figure 35a is considerably high. The explanation for this lies in the manner in which a student ReLU network can eliminate a redundant student. As mentioned in section 2, if for all input vectors $\boldsymbol{\xi}$ the activation is negative, the neuron will never be updated again. This situation can occur when a redundant student is being phased out. All incoming weights $\boldsymbol{W}_i$ to a neuron in the second hidden layer become negative over time and the activation will always be zero. These weight vectors may not have any form of alignment with the feedback weights, possibly even pointing in the opposite direction. This allows the angle for the alignment to be even higher than the same angle in overrealizable learning scenarios using erf networks.

5 Conclusion

In this thesis, we have discussed a new learning technique for neural networks called feedback alignment. This method can be used for learning in both shallow and deep feedforward neural networks. The technique differs from the widely used backpropagation, which uses identical and symmetric weights in the feedback stage. We compared the performance of feedback alignment to backpropagation using neural networks with the sigmoidal erf activation and networks with ReLU activation. Different learning scenarios were used to compare the performance of feedback alignment with backpropagation.

To answer the first research question, it appears that FA can perform at least as effectively as BP in most learning scenarios used in the experiments with erf networks. In realizable and overrealizable scenarios, the system changes the forward weights such that the learning process will resemble backpropagation and ϵ_g will reduce to zero. Neither method has a clear advantage in terms of the convergence rate. In overrealizable scenarios, we observed that the decrease of ϵ_g can change from exponential to power-law in deep networks for both methods. The reason this happens more often is deeper networks is likely due to the increased complexity. For a shallow network, is easier for the student to set the single components of $\boldsymbol{v}$ to zero. In deeper networks, the components in $\boldsymbol{W}$ have to be considered as well and students sharing specialization will happen at an increased frequency. This shared specialization in deep networks seems to occur more often when FA is used, possibly explaining why the decrease in ϵ_g in generally slower than BP when all plateau states are broken. In unrealizable scenarios, the rule cannot be realized and a zero generalization error cannot be reached. Both methods will converge to a system state in which ϵ_g is minimal.

The overall performance can differ quite substantially depending on the learning scenario. For particular initializations of the rule weights and student weights, the learning process can suffer from long symmetric states where the student has not specialized to the teacher. In relation to the second research question, using FA in the update steps can assist in breaking these unspecialized states in fewer discrete timesteps. The reason for this is unclear. However, we observed that when such a plateau state is reached, the alignment between the updates made by BP and those made by FA changes. When a loss plateau is reached, the angle $\Delta\mathbf{h}_{FA}\measuredangle\Delta\mathbf{h}_{BP}$ will almost exclusively increase. Essentially, a plateau is a fixed point in the learning dynamics, which is unstable. When the system is aligned, it behaves like backpropagation and when a fixed point is reached, the system has more degrees of freedom compared to BP as the weights are not constrained to be identical. This could be the reason that a plateau state is broken earlier. However, this does not explain the reason why the angle always increases, indicating that there is more theoretical understanding required of the learning dynamics when FA is used. There seems to be a correlation between the complexity of the student compared to the rule and the length and frequency of plateau states. In unrealizable scenarios, using FA has no particular positive effect on loss plateaus, possibly even increasing the length and frequency. For perfectly realizable scenarios, using FA might assist in earlier escape from a loss plateau. However, this does not appear to by systematic, as changing the initialization of the hidden layer weights for BP may achieve the same goal. In overrealizable scenarios, FA seems to have a fairly systematic advantage in terms of breaking plateau states at an earlier stage of learning, especially in the training of deeper networks.

In relation to the third research question, networks with ReLU activation can be trained using FA. However, a clear disadvantage is the probable need to use an overrealizable scenario, as the activation function is not symmetric and the feedback weights are fixed. However, in realistic machine learning situations, there are often more model parameters than there are training examples. Especially in deep learning, where networks can have millions of adaptable weights. Interestingly, as in the results for networks with erf activation, the breaking of plateau states in these overrealizable scenarios is assisted by using FA, most notably in the deeper networks.

6 Outlook

In this section, a number of possible future extensions and improvements to this thesis are outlined. These were not included in this project due to time or scope constraints.

6.1 Direct Feedback Alignment

In the paper by Arild Nøkland [9], a method is proposed where neurons receive teaching signals from disconnected paths. The error is propagated directly from the output layer to each hidden layer, using fixed random feedback connections. This novel principle, called direct feedback alignment, is another step towards biologically plausible machine learning techniques. Normal FA and BP require the error to travel backwards through symmetric or other reciprocal connections. This in itself is not biologically implausible, as cortical areas are known to be reciprocally connected [22]. However, the authors of [9] question whether this is how an error signal is relayed from one area of the brain to more distant areas. The feedback path becomes disconnected from the forward path, and the layer is no longer reciprocally connected to the layer above. For a 3-layer network with a single output unit, the gradients corresponding to the hidden layer update directions are

$$\boldsymbol{\delta a}_1 = \boldsymbol{b}_1 e \cdot \boldsymbol{g}'(\boldsymbol{a}_1),\ \boldsymbol{\delta a}_2 = \boldsymbol{b_2} e \cdot \boldsymbol{g}'(\boldsymbol{a}_2), \tag{39}$$

where $\boldsymbol{b}_i$ is a fixed random weight vector with appropriate dimension and $\boldsymbol{a}_i$ are the network activations at hidden layer i. If all hidden layers have the same number of hidden neurons, we can use the same random vector for each layer update. For example, if $N = 100$ and $K = M = 4$, we have a 100-4-4-1 network structure. This allows for the same feedback vector $\boldsymbol{b} \in \mathbb{R}^K$ to be used in the update steps for both $\boldsymbol{v} \in \mathbb{R}^K$ and $\boldsymbol{W} \in \mathbb{R}^{K \times K}$. For a short example experiment, overlaps are initialized with $Q_{ik} = 0$ for $i \neq k$, $Q_{ii} = 0.5$ and $R_{im} = 0$. Rule constants are isotropically initialized $T_{mm} = 1$ and $T_{mn} = 0$ for $m \neq n$. All feature vectors are sampled from a Gaussian distribution $\mathcal{N}(0, 1)$ and hidden units use the sigmoidal erf activation function. Figure 37 shows the evolution of ϵ_g for DFA and FA for a single run with identical initial conditions. For this particular initialization, using DFA results in an earlier escape from the loss plateau, following is a similar convergence rate for both methods. This simple example indicates that DFA can work at least as good as FA when the weights are updated using SGD. It might be an even better method in the context of breaking plateau states. However, in a future project, a more detailed comparison can be made between the effectiveness of the two variations of feedback alignment.

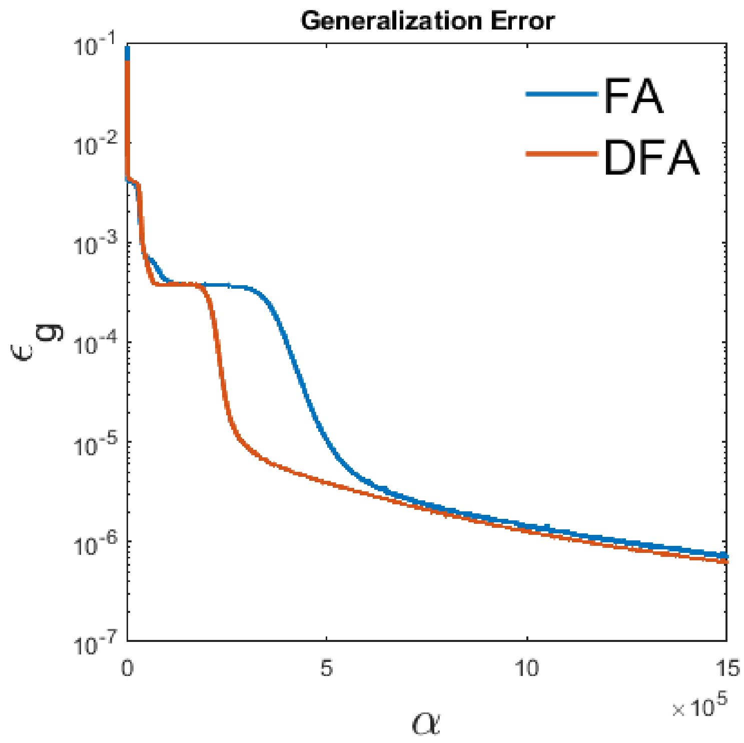


Figure 37: Evolution of ϵ_g for a single simulation of both FA and DFA with identical initial conditions. A 100-4-4-1 network with erf activation is used.

6.2 Efficient Computing Improvements

All experiments performed in this project were done on a single computer, using no parallelization or GPU speed-ups. This allows anyone to replicate these experiments without the need of any high-powered computing equipment. However, this could mean the simulations take a large amount of time, especially the experiments using deep networks. For a future project, it would be interesting to use parallelization to get a more accurate average from a larger number of simulations. Also, adding GPU computation for the purpose of doing many large matrix products in a fraction of the time it would take on the CPU. This would also allow for larger system sizes N to be studied.

6.3 Combination with Other Techniques

As mentioned, incorporating learning rate adaptation can speed up convergence. Also interesting would be including the effect of weight decay on the performance of FA. Different activation functions could be incorporated as well, like leaky ReLU [20] or the Piecewise Linear Unit(PLU) [23].

Dropout regularization [24] is a technique used to prevent overfitting in deep neural networks, when few examples are provided. This method randomly shuts units off from learning, which approximates training a large number of neural networks with different architectures in parallel. It would be interesting to study the effect of dropout regularization in combination with feedback alignment.

The rule network does not have to consist of fixed non-changing parameters. In a process known as *continual learning*, the target network can change over time and the learning system must be able to detect and track this *concept drift* [25, 26]. As the rule changes, intuitively one assumes that the alignment between the updates made by BP and those made by FA will also change. Therefore, the combination of FA and concept drift could be worthwhile topic of research.

6.4 Improved Theoretical Understanding

While the setup of the student-teacher scenarios is based on theoretical learning dynamics in large systems, the conclusions of this thesis are based almost entirely on empirical results. It would be useful to supplant these results with a better theoretical understanding of the learning dynamics when feedback alignment is used in conjunction with stochastic on-line gradient descent in multi-layer feedforward neural networks. This could provide an improved explanation as to why the the updates made feedback alignment de-aligns with those made by backpropagation in plateau states where the loss slows down. Moreover, if this de-alignment contributes to a faster escape from such a fixed point.

References

[1] D. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning representations by back-propagating errors”. In: *Nature* 323 (1986), pp. 533–536.

[2] Stork. “Is backpropagation biologically plausible?” In: *International 1989 Joint Conference on Neural Networks*. 1989, 241–246 vol.2. DOI: 10.1109/IJCNN.1989.118705.

[3] Ronald J. Williams. “Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning”. In: *Mach. Learn.* 8.3–4 (May 1992), 229–256. ISSN: 0885-6125. DOI: 10.1007/BF00992696. URL: https://doi.org/10.1007/BF00992696.

[4] Ruslan Salakhutdinov and Geoffrey Hinton. “Deep Boltzmann Machines”. In: *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*. Ed. by David van Dyk and Max Welling. Vol. 5. Proceedings of Machine Learning Research. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR, 2009, pp. 448–455. URL: https://proceedings.mlr.press/v5/salakhutdinov09a.html.

[5] Xiaohui Xie and H. Seung. “Equivalence of Backpropagation and Contrastive Hebbian Learning in a Layered Network”. In: *Neural Computation* 15 (2003), pp. 441–454.

[6] Dong-Hyun Lee et al. “Difference Target Propagation”. In: Aug. 2015, pp. 498–515. ISBN: 978-3-319-23527-1. DOI: 10.1007/978-3-319-23528-8_31.

[7] Yann Lecun. “Learning processes in an asymmetric threshold network”. English (US). In: *Disordered systems and biological organization, Les Houches, France*. Ed. by E. Bienenstock, F. Fogelman-Soulie, and G. Weisbuch. Springer-Verlag, 1986, pp. 233–240.

[8] Timothy P. Lillicrap et al. *Random feedback weights support learning in deep neural networks*. 2014. arXiv: 1411.0247 [q-bio.NC].

[9] Arild Nøkland. *Direct Feedback Alignment Provides Learning in Deep Neural Networks*. 2016. arXiv: 1609.01596 [stat.ML].

[10] P. Riegler. “Dynamics of on-line learning in neural networks”. In: 1997.

[11] M. Straat. “On-line learning in neural networks with ReLU activations”. In: 2018.

[12] Aleksandar Botev, Guy Lever, and David Barber. *Nesterov’s Accelerated Gradient and Momentum as approximations to Regularised Update Descent*. 2016. arXiv: 1607.01981 [stat.ML].

[13] Mark Ainsworth and Yeonjong Shin. “Plateau Phenomenon in Gradient Descent Training of ReLU networks: Explanation, Quantification and Avoidance”. In: *CoRR* abs/2007.07213 (2020). arXiv: 2007.07213. URL: https://arxiv.org/abs/2007.07213.

[14] G. Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of Control, Signals and Systems* 2 (1989), pp. 303–314.

[15] Frederieke Richert, Roman Worschech, and Bernd Rosenow. *Soft Mode in the Dynamics of Over-realizable On-line Learning for Soft Committee Machines.* 2021. arXiv: `2104.14546 [cond-mat.dis-nn]`.

[16] Silvia Scarpetta and David Saad. "On-line learning of unrealizable tasks". In: *Physical review. E, Statistical physics, plasmas, fluids, and related interdisciplinary topics* 60 (Dec. 1999), pp. 5902–11. DOI: `10.1103/PhysRevE.60.5902`.

[17] M Biehl and H Schwarze. "Learning by on-line gradient descent". In: *Journal of Physics A: Mathematical and General* 28.3 (1995), pp. 643–656. DOI: `10.1088/0305-4470/28/3/018`. URL: `https://doi.org/10.1088/0305-4470/28/3/018`.

[18] David Saad and Sara Solla. "On-line learning in soft committee machines". In: *Physical review. E, Statistical physics, plasmas, fluids, and related interdisciplinary topics* 52 (Nov. 1995), pp. 4225–4243. DOI: `10.1103/PhysRevE.52.4225`.

[19] David Bertoin et al. *Numerical influence of ReLU'(0) on backpropagation.* 2021. arXiv: `2106.12915 [cs.LG]`.

[20] Andrew L. Maas. "Rectifier Nonlinearities Improve Neural Network Acoustic Models". In: 2013.

[21] Michael Biehl, Peter Riegler, and Christian Wöhler. "Transient dynamics of on-line learning in two-layered neural networks". In: *Journal of Physics A: Mathematical and General* 29 (Aug. 1996), pp. 4769–4780. DOI: `10.1088/0305-4470/29/16/005`.

[22] Charles Gilbert and Wu Li. "Top-down influences on visual processing". In: *Nature reviews. Neuroscience* 14 (Apr. 2013), pp. 350–363. DOI: `10.1038/nrn3476`.

[23] Andrei Nicolae. "PLU: The Piecewise Linear Unit Activation Function". In: *CoRR* abs/1809.09534 (2018). arXiv: `1809.09534`. URL: `http://arxiv.org/abs/1809.09534`.

[24] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15 (June 2014), pp. 1929–1958.

[25] Michiel Straat et al. "Statistical Mechanics of On-Line Learning Under Concept Drift". English. In: *Entropy* 20.10 (Oct. 2018). ISSN: 1099-4300. DOI: `10.3390/e20100775`.

[26] M. Straat et al. "Supervised learning in the presence of concept drift: a modelling framework". In: *Neural Computing and Applications* (2021). ISSN: 1433-3058. DOI: `10.1007/s00521-021-06035-1`. URL: `http://dx.doi.org/10.1007/s00521-021-06035-1`.

A Extended Methods and Initializations

This appendix section describes a number of extended explanations regarding the methods used and possible issues regarding the initializations and their solutions.

A.1 Initialization in Narrow Networks

Because we study the performance of FA in relatively networks with low width, we may have to initialize our hidden-to-hidden and hidden-to-output weights in such a manner that it represents wider networks. For example, in the thermodynamic limit, $N \to \infty$, any set of feature vectors sampled from a normal distribution $\mathcal{N}(0,1)$, will also be normally distributed, i.e. have 0 mean and 1 variance. This ensures that the magnitude of these feature vectors will be significant enough. If we sample only few parameters from $\mathcal{N}(0,1)$, these properties are not guaranteed. This can hinder the performance in a student-teacher scenario. In a 3-layer network, if some columns (weights coming from a node) of the hidden-to-hidden weight matrix W for the rule network consist of small values, the student network might see this node as not significant. The weights coming from that node may be set to zero. This does not allow the student to perfectly represent the rule, the training process will converge to a wrong local minimum and a zero generalization error will not be achieved. Therefore, we can scale the columns $\boldsymbol{W}_j^*$, of the teacher network, such that their magnitude is high enough. Using a scaling factor f, we can scale every $\boldsymbol{W}_j^*$ as follows,

$$W_j = \frac{W_j}{||W_j||f}. \tag{40}$$

For example, if f is 1, this scaling forces all W_j to have unit norm. Obviously, each vector component of the hidden-to-output weights $\boldsymbol{v}^*$ is just a single value, a column with length 1. For these values, we can exclude numbers below a certain threshold. As the feedback weights are fixed, we want to perform a similar initialization process for $\boldsymbol{B}$ and $\boldsymbol{b}$. Ensuring that each row of $\boldsymbol{B}$ has significant magnitudes. Not doing this can significantly slow down learning.

A.2 Alignment Calculation

In section 3, it was shortly explained how the alignment between FA and BP is calculated for a 2-layer network. As the inverse cosine alignment between $\boldsymbol{b}$ and $\boldsymbol{v}$,

$$\theta_1 = cos^{-1}\frac{\boldsymbol{b}\cdot\boldsymbol{v}}{||\boldsymbol{b}||||\boldsymbol{v}||},$$

as the inverse cosine of two non-zero vectors of an inner-product space. The result of $\frac{\boldsymbol{b}\cdot\boldsymbol{v}}{||\boldsymbol{b}||*||\boldsymbol{v}||}$ lies in the range $[-1,1]$. If $\theta_1 = -1$, $\boldsymbol{b}$ and $\boldsymbol{v}$ point in the opposite direction of each other. If $\theta_1 = 1$, they point in the same direction. If θ_1 is 0, the two vectors are orthogonal and completely uncorrelated. The inverse cosine converts this to an angle in radials, between 0 and π. Converting this to an angle in degrees, we get an alignment between 0 and 180 degrees. This angle represents exactly the angle between the hidden-unit updates for FA and of that prescribed by BP. If a 3-layer network is used, we need also the angle between $\boldsymbol{W}^T$ and $\boldsymbol{B}$, We can easily get the average angle of all vectors in $\boldsymbol{W}^T$ and $\boldsymbol{B}$ by vectorizing both matrices. A vectorization is a linear transformation of a matrix to a column vector. This column vector is obtained by stacking the column of a matrix on top of one another. Vectorization of $\boldsymbol{W}^T$ and $\boldsymbol{B}$ results in two long vectors $\boldsymbol{w}_v$ and $\boldsymbol{b}_v$. Again, we take the inverse cosine alignment between these two vectors

$$\theta_2 = cos^{-1}\frac{\boldsymbol{b}_v \cdot \boldsymbol{w}_v}{||\boldsymbol{b}_v||\ ||\boldsymbol{w}_v||}. \tag{41}$$

The total alignment between the updates for FA and BP for both layers is the average over the angle of both layers.

$$\theta = \frac{\theta_1 + \theta_2}{2}. \tag{42}$$

B Additional Results

This appendix section showcases a number of additional results related to the experiments in section 3 and 4 in no particular order. These results were not shown in these sections because their relevancy was not considered important enough to contribute to the conclusions made in this thesis.

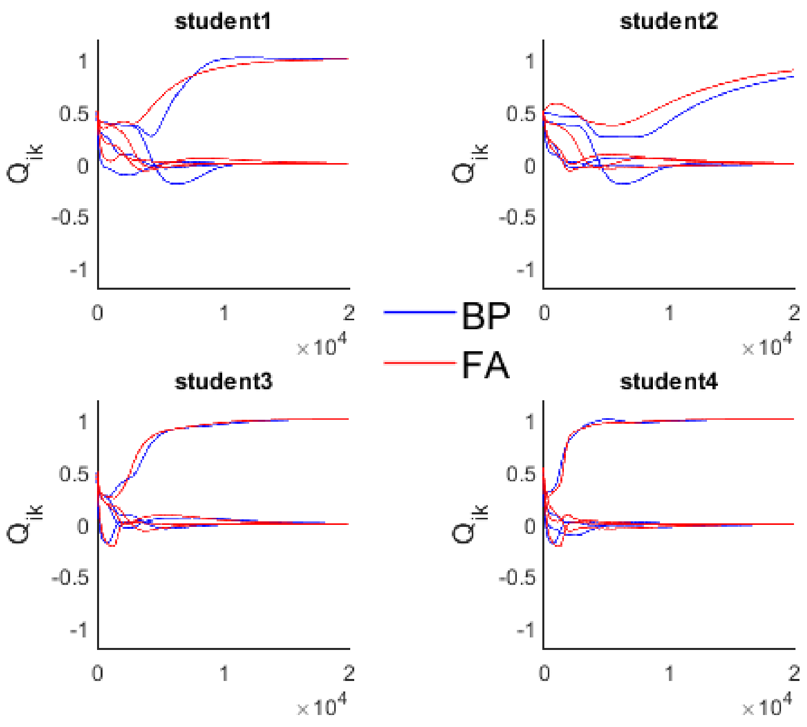


Figure 38: Evolutino of the student-student overlaps Q_{ik} for a 500-4-1 student and teacher network with erf activation. After initial symmetry, all students gain zero overlap with other students. Their magnitude finalizes at 1, same as all teacher vectors, implying that a student has either fully specialized or anti-specialized to a teacher. As the results sections already contained a larger number of graphs, the Q_{ik} graphs were not shown in there. Moreover, they usually directly correspond with the R_{im} graphs, depending on the realizability of a scenario.

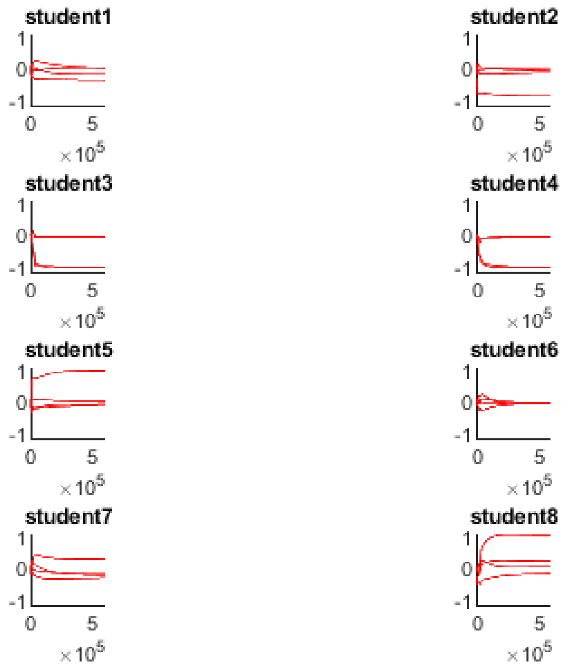


Figure 39: Order parameters R_{im} in a situation where the decrease of ϵ_g goes from exponential to power-law, as in Figure 30a. Here, multiple students have some form of shared specialization to a teacher

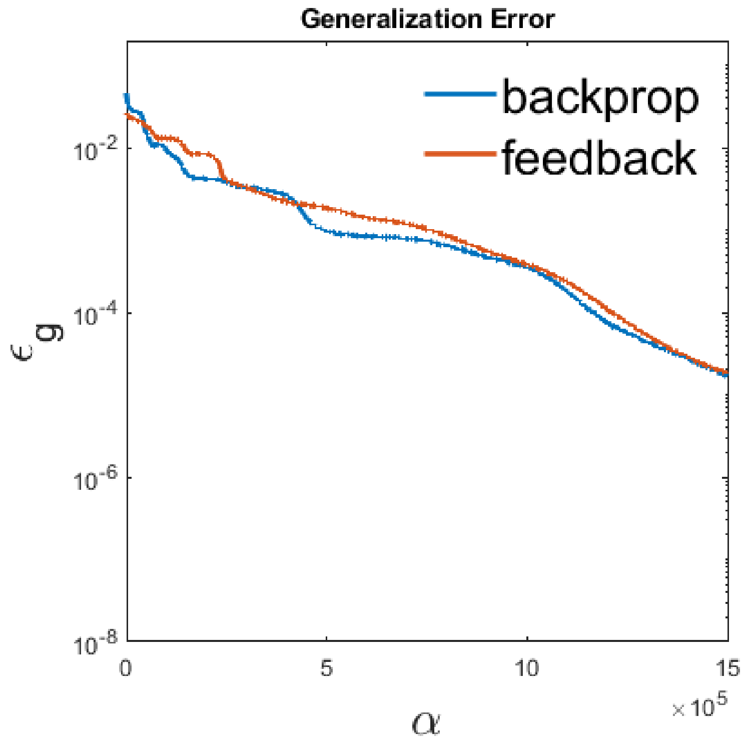


Figure 40: Evolution of ϵ_g for a 100-10-5-1 student and teacher network with erf activation for both BP and FA with identical initial conditions. This graph is shown here as an indication that the proficiency of FA to break plateau states in overrealizable scenarios shown in section 3 and 4 is due to the increased complexity of the student compared to the rule. Instead of the reason being that FA performs better when the width of the hidden layers are larger in general. Of course, more simulations are needed to draw any decisive conclusion, which is why this graph was not included in the experimental result sections

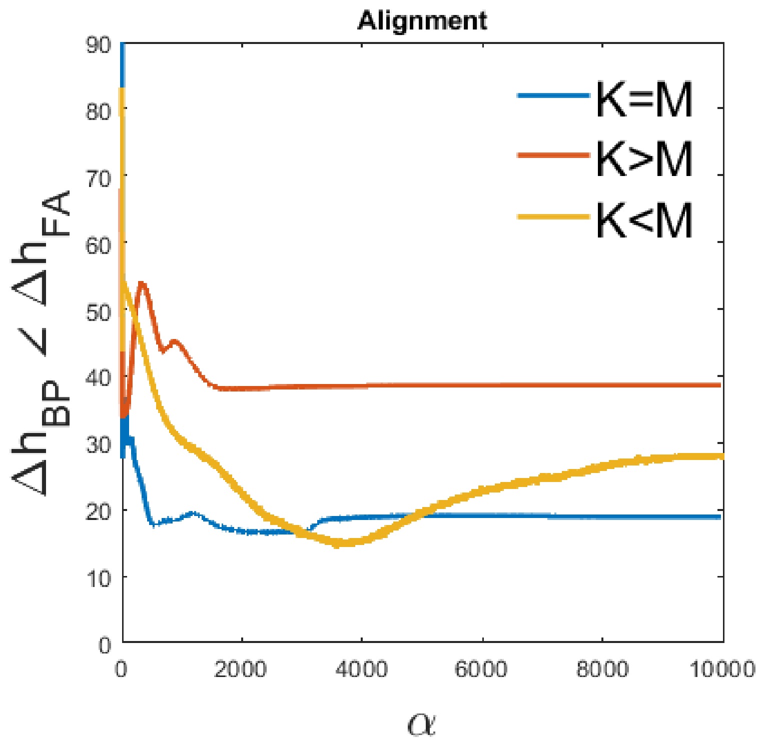


Figure 41: The average angle between the updates made by BP and FA for shallow sigmoidal erf networks for $n = 10$ simulations for each scenario. This graph showcases that when the scenario is perfectly realizable, the angle is lowest. For overrealizable scenarios the angle is highest.