Predictive Coding and Biologically Plausible Neural Networks

Bachelor Project

Anders Bredgaard Thuesen
s183926@student.dtu.dk

January 2022

Abstract

Contents

1 Introduction

1.1 Motivation

In the recent years, deep learning has shown impressive results due to the availability of massive parallel compute and huge amounts of data. From the biological inspiration of the neuron to the perceptron where data inputs are weighted, summed together and thresholded, several new modern architectures, like recurrent, residual and transformer neural networks have pushed the limits and achieved state of the art results in speech recognition, computer vision and natural language understanding. Despite of these networks being originally inspired by the brain, the backpropagation (backprop) algorithm for learning the weights and deep learning in general has been criticized for being biologically implausible [1]. This project will primarily be dealing with on of them: The weight transport problem which arises from the way backprop uses the connection weights in both the forward pass (inference) and the backwards parse (calculating the gradients), requiring that both forward and backward connections have symmetric weights and that information is able to flow backwards through the weights.
Besides having both philosophical as well scientific interest, studying the computational aspects of how the human brain processes sensory input might lead to great improvements in deep learning and artificial intelligence.

1.2 Related work

Several attempts has been made to make modern deep learning more biologically plausible. These can be divided into two types of categories. The first category consists of methods that try to optimize the inference on low-powered neuromorphic hardware such as the Intel Loihi or IBM TrueNorth chips, by converting existing neural network architectures into their spiking counterparts. The other category consists of methods that aim to make the learning phase biologically plausible by only relying on local weight updates in order to optimize for some objective. This is in alignment with the Hebbian learning theory from the neuroscience litterature which states that the synaptic plasticity is only dependent on the pre- and post-synaptic activity, possibly modulated through some global signaling mechasnism (ex. dopamine). One example hereof is the work by Bengio et al. on Continual Equilibrium Propagation [2].

1.3 Research questions

The project will address the following three research questions:

- According to the current literature, what are the biological constraints of biological learning?
- To what extent can predictive coding be used to approximate backpropagation under the above mentioned biological constraints?
- In what ways can modern deep learning benefit from biological plausible learning algorithms?

2 Classical deep learning

2.1 Dataset

We define our dataset, $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}$ for $i = 1 \dots N$ consisting of N pairs of datapoints, $\mathbf{x}_i \in \mathbb{R}^k$ and $\mathbf{y}_i \in \mathbb{R}^l$ where N is the size of the dataset. We notice, that both $\mathbf{x}_i$ and $\mathbf{y}_i$ can be vectors of possibly differently dimensions k and l respectively. Here $\mathbf{x}_i$ might represent eg. an image with its horizontal and channel axes flattened into a single one-dimensional vector, as we will encounter later when training on images of handwritten digits in the MNIST dataset [4]. In the case of supervised learning, our objective is from $\mathbf{x}_i$ to predict the corresponding label, $\mathbf{y}_i$ which amounts to a single scalar number from 0 to 9 in the MNIST dataset.

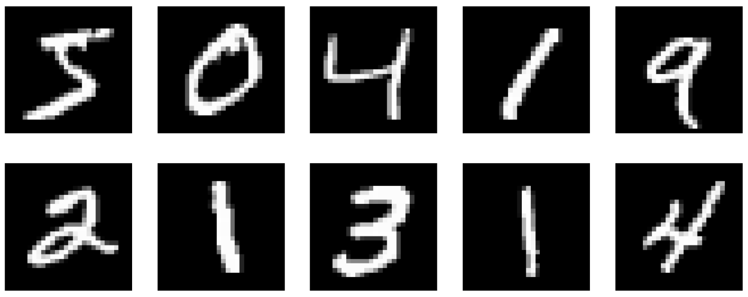
Figure 1: First 10 examples of the MNIST training dataset.

2.2 Feedforward neural networks

Feedforward neural networks (FNN) are considered the simplest kind of neural network where the connections between the nodes does not allow for any cycles or recurrent connections. Feedforward neural networks are divided into several layers, where input data from the first layer is "feed forward" through the so-called hidden layers to the final output layer of the network, considered the prediction of the network. FNNs are typically fully connected networks which entails that every node of the network is connected to every node in the previous layer. One special case of FNNs is that of when there are no hidden layers in the network and the input layer is linearly transformed to the output layer and "activated" through a softmax or sigmoid function, depending on the output of output nodes. In that case, the FNN will correspond to (multinomial) logistic regression. This correspondance incentivises the use of activation functions after each linear transformation of the layers, as the network would otherwise not be able to describe non-linearities in the data. Historically, the sigmoid activation function $\sigma(z) = (1 + \exp(-z))^{-1}$ has been the go-to activation function, but recently the rectified linear unit, $\mathrm{ReLU}(z) = \max(0, z)$, has become the de facto standard.

A feedforward neural network with $L - 2$ hidden layers is parametarized by the weight matrices $\mathbf{W}^{(l)} \in \mathbb{R}^{m \times n}$ and biases $\mathbf{b}^{(l)} \in \mathbb{R}^m$ for $l = 2 \dots L$ where n is the input dimension of the layer and m the output dimension. A feedforward pass from layer $l-1$ to layer l is given by $\mathbf{a}^{(l)} = \sigma(\mathbf{z}^l)$ where $\mathbf{z}^{(l)} = \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$ and σ is the chosen activation function. By letting the initial activation $\mathbf{a}^{(1)} = \mathbf{x}_i$ one can consider the final activation of the network the prediction of the network $\hat{\mathbf{y}}_i = \mathbf{a}^{(L)}$.

We initialize the weight matrices using Kaiming He initialization, where the entries of the matrix $W_{ij}^{(l)}$ are drawn from a normal distribution with zero mean and $\sqrt{2/n}$ standard deviation as this will help reduce vanishing and exploding gradient problem by keeping the variance in each layer equal when using ReLU activation. [3]

2.3 Back-propagation

The working horse of allmost all modern deep learning models is the back-propagation (aka. backprop) algorithm first popularized for training neural networks by Rumelhart, Hinton & Williams in 1986 [6]. The algorithms solves what is referred to as the *credit-assignment problem*. When learning the parameters of an artificial neural network we would like to know how changing a weight in the network contributes to the total loss, in order to change it in the direction that minimizes the loss. One naive way to do this would be simply to adjust a single random weight slightly, evaluate the new neural network on the dataset and observe the effect on the model loss. If the change leads to a decrease in loss, keep the change, otherwise repeat from the beginning. This would however be very computationally expensive, since the network would have to be evaluated on the entire dataset for each weight in the network. Fortunately, the backprop algorithm achieves this much more efficiently as we will see in the following section.

Back-propagation is an efficient method for calculating the weight updates that minimizes some loss function, $\mathcal{L}(\hat{\mathbf{y}}_i, \mathbf{y}_i)$, which measures the difference between the predicted output of the network, $\hat{\mathbf{y}}_i$, and the true output, $\mathbf{y}_i$. Examples of loss functions are squared error $\sum \frac{1}{2}(\hat{\mathbf{y}}_i - \mathbf{y}_i)^2$, typically used for regression and categorical cross entropy $-\sum \mathbf{y}_i \cdot \log(\hat{\mathbf{y}}_i)$ for classification. At its core, the back-propagation algorithm is simply applying the chain rule on the partial derivate of the loss function with respect to the parameters and realizing that a lot of computation can be reused or "back propagated" in order to calculate the weight and bias updates for earlier layers. It is therefore necessary that the loss function is differentiable with respect to the prediction variable. Some alternatives to back-propagation exist such as Direct Feedback Alignment [5], but is not commonly used in practice.

To demonstrate the efficiency of the back-propagation algorithm, one can consider the last and second to last layers of the network. Applying the chain rule on the partial derivative of the loss function wrt. $\mathbf{W}^{(L)}$ yields

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(L)}} = \underbrace{\frac{\partial \mathcal{L}}{\partial \mathbf{a}^{(L)}} \frac{\partial \mathbf{a}^{(L)}}{\partial \mathbf{z}^{(L)}}}_{\delta^{(L)}} \frac{\partial \mathbf{z}^{(L)}}{\partial \mathbf{W}^{(L)}} = \underbrace{\mathcal{L}'(\hat{\mathbf{y}}_i)\sigma'(\mathbf{z}^{(L)})}_{\delta^{(L)}} \mathbf{a}^{(L-1)} \tag{1}$$

whose factors can be divided into the error term, $\delta^{(L)}$, and activation in the previous layer, $\mathbf{a}^{(L-1)}$. Yet again applying the chain rule on the partial derivative of the loss function, but this time wrt. $\mathbf{W}^{(L-1)}$ hints a the source of efficiency:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(L-1)}} = \overbrace{\frac{\partial \mathcal{L}}{\partial \mathbf{a}^{(L)}} \frac{\partial \mathbf{a}^{(L)}}{\partial \mathbf{z}^{(L)}}}^{\delta^{(L)}} \frac{\partial \mathbf{z}^{(L)}}{\partial \mathbf{a}^{(L-1)}} \frac{\partial \mathbf{a}^{(L-1)}}{\partial \mathbf{z}^{(L-1)}} \frac{\partial \mathbf{z}^{(L-1)}}{\partial \mathbf{W}^{(L-1)}} = \underbrace{\delta^{(L)} \mathbf{W}^{(L)} \sigma'(\mathbf{z}^{(L-1)})}_{\delta^{(L-1)}} \mathbf{a}^{(L-2)} \tag{2}$$

Though the equation above only considers $\mathbf{W}^{(L)}$ and $\mathbf{W}^{(L-1)}$ it can be shown in general that

$$\delta^{(L)} = \mathcal{L}'(\hat{\mathbf{y}}_i)\sigma'(\mathbf{z}^{(L)}), \quad \delta^{(l)} = \delta^{(l+1)}\mathbf{W}^{(l+1)}\sigma'(\mathbf{z}^{(l)}) \tag{3}$$

$$\mathbf{W}^{(l)} = \delta^{(l)}\mathbf{a}^{(l-1)} \tag{4}$$

We can then update our parameters as such.

$$\mathbf{W}^{(l)} = \mathbf{W}^{(l)} - \alpha\frac{\partial\mathcal{L}}{\partial\mathbf{W}^{(l)}} \tag{5}$$

$$\mathbf{b}^{(l)} = \mathbf{b}^{(l)} - \alpha\frac{\partial\mathcal{L}}{\partial\mathbf{b}^{(l)}} \tag{6}$$

3 Biologically plausible deep learning

3.1 Biological constraints

Write something about biological constraints here.

3.2 Spiking Neural Networks

Define spiking neural networks here. Why are they cool?

3.2.1 LIF Neuron model

What are LIF neurons? What are their dynamics?

3.2.2 ANN to SNN conversion

How do we convert an ANN to a SNN and run the stuff live?

3.3 Predictive Coding and Z-IL

What is predictive coding? Mention Rao and Ballard. How can they calculate gradients?

3.3.1 Equivalence to back-prop

3.3.2 Variational free energy

3.4 Variational Inference

3.5 Energy Based Models

4 Results

5 Discussion

6 Conclusion

References

[1] Yoshua Bengio, Dong-Hyun Lee, Jörg Bornschein, and Zhouhan Lin. Towards biologically plausible deep learning. *CoRR*, abs/1502.04156, 2015.

[2] Maxence Ernoult, Julie Grollier, Damien Querlioz, Yoshua Bengio, and Benjamin Scellier. Equilibrium propagation with continual weight updates. *CoRR*, abs/2005.04168, 2020.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. 2015.

[4] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[5] Arild Nøkland. Direct feedback alignment provides learning in deep neural networks, 2016.

[6] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.