Overview: Backpropagation and Predictive Coding

Samuel Lippl

April 23, 2018

Contents

1 Backpropagation

1.1 We consider an additive feed-forward neural network in layered structure $\theta \in \mathbb{R}^{\mathcal{V}\times\mathcal{V}}_{c,v}$.

1 Open Problem (Layered structure). Construction of layers and the corresponding order on $\mathcal{V}$.

The step function is therefore

$$T^s_\theta = Wf \rhd s \tag{1}$$

and the canonical network function is

$$\mathbb{R}^{\mathcal{V}^{(0)}} \ni i \mapsto N^i_\theta = Wf \rhd i \rfloor \in \mathbb{R}^{\mathcal{V}} \tag{2}$$

where we denote the corresponding output function by

$$O^i_\theta \in \mathbb{R}^{\mathcal{V}^{(L)}} \tag{3}$$

We will often leave θ and i implicit where they are clear from context.

1.2 (Backpropagation Objectve Function) The backpropagation algorithm is concerned with optimizing the prediction that is generated by a neural network with respect to some loss function. We will first consider the quadratic loss function between a prediction O generated from some input and the real output $o \in \mathbb{R}^{\mathcal{V}^{(L)}}$[1]

$$E^{i;o}_\theta := E(O^i_\theta, o) := \frac{1}{2}(O^i_\theta - o)^T(O^i_\theta - o) \tag{4}$$

[1] The given framework is easily extendable to a training set with more than one observation.

θ is therefore modified to reduce E which we do by gradient descent:

$$\theta \mapsto \theta + \alpha \frac{\partial E}{\partial \theta} \tag{5}$$

As the loss function is generally one-dimensional, we can consider $\frac{\partial E}{\partial \theta}$ to be analogous to θ.

1.3 Theorem (Backpropagation Identity). Defining

$$\delta_{\text{diff}} := O_\theta^i - o \tag{6}$$

and

$$\delta := f'(N) \circ (\theta \delta_{\text{diff}} \restriction) \tag{7}$$

we obtain the following identity:

$$\frac{\partial E}{\partial \theta} = \delta f(N)^T \tag{8}$$

Proof. (8) follows from a repeated application of the chain rule. We first prove that $\delta = \frac{\partial E}{\partial N}$. We start by observing that

$$\frac{\partial E}{\partial N(\mathcal{V}^{(L)})} = \frac{\partial E}{\partial O} = \delta_{\text{diff}} \tag{9}$$

For $v \notin \mathcal{V}^{(L)}$, we can write recursively (as we have a feed-forward network)

$$\frac{\partial E}{\partial N(v)} = \frac{\partial E}{\partial N(v \to)} \frac{\partial N(v \to)}{\partial N(v)} = \frac{\partial E}{\partial N(v \to)} \theta_{v \to}(v \to) \cdot f'(N(v \to)) \tag{10}$$

As all other entries in $\theta_{v \to}$ are 0_ς, this corresponds to the recursive definition in (7)[2].
As

$$\frac{\partial N(w)}{\partial \theta_{v \to w}} = f(w) \tag{11}$$

and therefore

$$\frac{\partial E}{\partial \theta_{v \to w}} = \delta(v) f(w) \tag{12}$$

we obtain (8) which proves the theorem. □

1.4 Remark. Backpropagation is only defined on feedforward networks, where the proof of theorem 1.3 demonstrates the point at which a recurrent network would fail the algorithm. While we can also formulate the backpropagation algorithm for non-additive neural networks, we do not win much by that. It is the particular additive structure that allows the concise recursive notation.

2 Predictive Coding

I will present the predictive coding model according to [4] in a generalized framework.

2.1 (Generative Model) We first consider a so-called *generative model* $\xi \in \mathbb{R}^{\mathcal{V}_{\text{gen}} \times \mathcal{V}_{\text{gen}}}$ as a neural network which is a stochastic additive feed-forward network that is developed depending on input states $s(\mathcal{V}_{\text{gen}}^{(0)})$ by

$$s(v) = \mu_v\left(s(\to v)\right) + \epsilon_v \qquad \epsilon \sim \mathcal{N}(0, \Sigma) \tag{13}$$

where $\Sigma \in \mathbb{R}^{\mathcal{V}_{\text{gen}}}$ is a positive definite covariance matrix and the expected value is

$$\mu_v\left(s(\to v)\right) = \xi_{\to v}^T s(\to v) \tag{14}$$

[2] The multiplication by $f'(N)$ does not have to be part of the recursive definition.

2.2 Remark. Σ is often a diagonal matrix or only contains correlations across the same layers.

2.3 (Objective Function of Predictive Coding) While backpropagation is focused on optimizing the prediction, predictive coding attempts to find the most likely state of the entire generative model. As is often case, we consider the log-likelihood and our objective function is

$$F_{\xi,\Sigma}(s) := \ln P\left(s\left(\mathcal{V}\setminus\mathcal{V}^{(0)}_{\text{gen}}\right)|s\left(\mathcal{V}^{(0)}_{\text{gen}}\right)\right) \tag{15}$$

which we obtain in proposition2.4.

2.4 Proposition. Optimizing F corresponds to optimizing

$$-\frac{1}{2}\left(\ln\left(\det\Sigma\right)+\epsilon^T\Sigma^{-1}\epsilon\right) \tag{16}$$

where we have only ommitted a constant term. We therefore denote (16) by F.

Proof. As $\epsilon\in\mathbb{R}^{\mathcal{V}\setminus\mathcal{V}^{(0)}}$,

$$F_{\xi,\Sigma}(s)=\ln p\left(\epsilon|0,\Sigma\right) \tag{17}$$

where p is the multivariate normal distribution with expectation 0 and covariance Σ:

$$p(\epsilon|0,\Sigma)=\frac{1}{\sqrt{(2\pi)^{|\mathcal{V}|}\det\Sigma}}\exp\left(-\frac{1}{2}\epsilon^T\Sigma^{-1}\epsilon\right) \tag{18}$$

Note that F in the form of (17) still depends on s as

$$\epsilon=s-\xi^T f(s) \tag{19}$$

where we omit all $v\in\mathcal{V}^{(L)}$ as their error does not behave stochastically but is trivially zero. Logarithmizing (18) yields

$$\ln p=-\ln\left(\sqrt{(2\pi)^{|\mathcal{V}|}}\right)-\ln\left(\sqrt{\det\Sigma}\right)-\frac{1}{2}\epsilon^T\Sigma^{-1}\epsilon \tag{20}$$

where we omit the constant term which, together with some basic transformations yields (16). □

2.5 (Inference) Clearly, if we are are given $s(\mathcal{V}^{(0)})$, the F-optimal prediction of all other states would be

$$\theta^T s(\mathcal{V}^{(0)})\downharpoonright \tag{21}$$

However, we generally cannot observe $\mathcal{V}^{(0)}$ which corresponds to a cause that may be an abstract concept such as a face and therefore even constructed by our brain. In the general framework, there is a subset $\mathcal{I}\subseteq\mathcal{V}$ of which we know the value and have to infer the state of the remaining network ξ by optimizing F over the PUs $\mathcal{V}\setminus\mathcal{I}=:\mathcal{H}$.

2.6 Theorem (Predictive Coding Model I: Prediction). The neural network with the step function on $s\in\mathbb{R}^{\mathcal{V}\times\mathcal{E}}$

$$T^{(s,\varepsilon)}_{\xi}(\mathcal{V})=s-\varepsilon+\operatorname{diag}\left(f'(s)\right)\xi\varepsilon \tag{22}$$

$$T^{(s,\varepsilon)}_{\xi}(\mathcal{E})=\Sigma^{-1}\left(s-\xi^T f(s)\right) \tag{23}$$

performs gradient descent on F. By leaving $\mathcal{I}$ constant, we may perform inference according to 2.5.

Proof. We observe that

$$\frac{\partial F}{\partial N} = \frac{\partial F}{\partial \varepsilon}\frac{\partial \varepsilon}{\partial N} \tag{24}$$

and, as

$$\varepsilon = \Sigma^{-1}\epsilon$$

we can write

$$F = -\frac{1}{2}\varepsilon^T \Sigma \varepsilon \tag{25}$$

Therefore

$$\frac{\partial F}{\partial \varepsilon} = -\varepsilon^T \Sigma \tag{26}$$

and

$$\frac{\partial \varepsilon}{\partial N} = \Sigma^{-1}\left(I - \xi^T \mathrm{diag}(f'(N))\right) \tag{27}$$

where $I \in \mathbb{R}^{\mathcal{V}\times\mathcal{V}}$ is the identity matrix and $J = \mathrm{diag}(f'(N))$ is the diagonal matrix where $J_{v\to v} = f'(N(v))$. We can now put together (24,26,27) to obtain (22) □

2.7 (Learnable Σ) Theorem 2.6 provides an implicit definition of a neural network which is neither additive nor feed-forward.

2 Open Problem (Connection weights as learnable parameters). I am still missing the section where I give a reason for the following.

Additionally, we would like to treat Σ as learnable parameters and therefore want to identify them with connection weights. The following definition provides a neural network with a weight matrix that contains all learnable parameters. It is important to note that while the network is not additive, it would be relatively easy to render the network additive by defining additional nodes for f and f'. I do not see the point of such a definition as the network in theorem 2.10 is biologically plausible according to [4].

2.8 Definition (Predictive Coding Model II: Predictive Network). If we define error nodes for all PUs but the first layer of the generative model

$$\varepsilon := \left\{\varepsilon_v | v \in \mathcal{V}_{\text{gen}} \setminus \mathcal{V}^{(0)}_{\text{gen}}\right\} \tag{28}$$

the matrices

$$I_{\mathcal{V}} = \begin{pmatrix} 1_\varsigma & & \\ & \ddots & \\ & & 1_\varsigma \end{pmatrix} \in \mathbb{R}^{\mathcal{V}\times\mathcal{V}}_{c,v}, I_\varepsilon = \begin{pmatrix} 1_\varsigma & & \\ & \ddots & \\ & & 1_\varsigma \end{pmatrix} \in \mathbb{R}^{\varepsilon\times\varepsilon}_{c,v} \tag{29}$$

the Processing Units

$$\mathcal{V} = \mathcal{V}_{\text{gen}} \cup \varepsilon \tag{30}$$

and $\xi^\varepsilon \in \mathbb{R}^{\varepsilon\times\mathcal{V}_{\text{gen}}}$ (resp. $I'_{\mathcal{V}} \in \mathbb{R}^{\varepsilon\times\mathcal{V}_{\text{gen}}}$) as the weight matrix ξ (resp. $I_{\mathcal{V}}$) without the rows of the input layer $\mathcal{V}^{(0)}$, the *predictive coding network* is defined by

$$\theta = \begin{pmatrix} I_{\mathcal{V}} & -\xi^T + I'^T_{\mathcal{V}} \\ -I'_{\mathcal{V}} + \xi & (I_\varepsilon - \Sigma)^T \end{pmatrix} \in \mathbb{R}^{\mathcal{V}\times\mathcal{V}} \tag{31}$$

together with the unit functions

$$g_v(s|\theta_{\to v}) := \theta^T_{\to v}(\mathcal{V})s(\mathcal{V}) + \theta^T_{\to v}(\varepsilon)s(\varepsilon)f'(N(v)) \tag{32}$$

$$g_{\varepsilon_v}(s|\theta_{\to \varepsilon_v}) := \theta^T_{\to \varepsilon_v}(\mathcal{V})f(s(\mathcal{V})) - f(s(v)) + s(v) + \theta^T_{\to \varepsilon_v}(\varepsilon)s(\varepsilon) \tag{33}$$

2.9 Remark. By expressing (32,33) by ξ and Σ instead of θ, their definition and the similarity to (22,23) becomes clear:

$$g_v(s|\theta_{\to v}) = s(v) - s(\varepsilon) + f'(s(v))\xi^{\varepsilon}_{v\to} s(\varepsilon) \tag{34}$$

$$g_v(s|\theta_{\to \varepsilon_v}) = s(v) - (\xi^{\varepsilon}_{\to v})^T s(\mathcal{V}) + (I_\varepsilon - \Sigma)s(\varepsilon) \tag{35}$$

2.10 Theorem. The predictive coding network has the local maxima of F as fixed states.

Proof. It is sufficient to prove that the fixed states of θ are identical to the fixed states of the step function defined in (22,23). We use the equations from remark 2.9 where (34) implies (22). For (35), we observe that for some fixed state t,

$$I_\varepsilon t(\varepsilon) = t(\varepsilon) = t(v) - (\xi^{\varepsilon}_{\to v})^T f(t(\mathcal{V})) + (I_\varepsilon - \Sigma)t(\varepsilon)$$

which is, again, equivalent to the fixed state equation corresponding to 23. □

2.11 (Predictive Coding Model III: Learning Perspectives) Theorem 2.10 can be taken as evidence that the predictive coding network is a reasonable implementation of the generative model in the brain. We will now turn towards the question how such a network may learn its parameters. With respect to this goal, we have two options: we may consider learning ξ and Σ or we may consider learning θ. While every learning rule on ξ and Σ can be transformed into a learning rule on θ this transformation does not preserve the Hebbian property because of the two applications of ξ, a problem that has been coined *weight symmetry* (see e. g. [4, p. 1254]. On the other hand, not every learning rule on θ can be transformed into a learning rule on ξ and Σ but if there exists such a transformation, it preserves the Hebbian property.

2.12 (Predictive Coding Model IV: Learning ξ and Σ) In this framework, we begin by treating some fixed state of the network as a constant value and try to improve ξ and Σ, again by gradient descent. This corresponds to the approach in [4] and is based on the expectation maximization algorithm [2] as presented in [3]. In this framework, prediction corresponds to the estimation of the expected value of all units and therefore the states of the generative model if we fix the states of certain units. On the other hand, learning ξ and Σ corresponds to changes in the generative model the brain builds to describe its environment (see [3]).

2.13 Theorem (Learning ξ). If

$$\frac{\partial F}{\partial \xi^\varepsilon} \equiv \left(\frac{\partial F}{\partial \xi_{v\to w}}\right)_{(v,w)\in\mathcal{V}\times(\mathcal{V}\setminus\mathcal{V}^{(0)})} \tag{36}$$

we obtain

$$\frac{\partial F}{\partial \xi^\varepsilon} = N(\varepsilon) f(N(\mathcal{V}))^T \tag{37}$$

Proof.

$$\frac{\partial F}{\partial \xi_{v\to w}} = -\varepsilon^T \Sigma \frac{\partial \varepsilon}{\partial \xi_{v\to w}}$$

$$\frac{\partial \varepsilon}{\partial \xi_{v\to w}} = -\Sigma^{-1} \frac{\partial}{\partial \xi_{v\to w}} \xi^T f(N)$$

As $\xi^T f(N)$ contains $\xi_{v\to w}$ exactly once, we have a vector with one non-zero entry. As $\xi_{v\to w}$ is in the w-row of ξ^T, it is in the w-row of $\xi^T f(N)$. Evidently, the value in the w-row is $f(N(v))$ and therefore

$$\frac{\partial F}{\partial \xi_{v\to w}} = \varepsilon_w f(N(v)) \tag{38}$$

which, in the notation of (36), yields (37). □

2.14 (Hebbian Learning of ξ) We note that ξ can be learned in a Hebbian way that can be directly motivated from gradient ascent. We also note that the formula resembles the formula for backpropagation but is generally not identical because the network functions may yield different results.

2.15 Proposition (Learning Σ I: Gradient Ascent). If

$$\frac{\partial F}{\partial \Sigma} \equiv \left(\frac{\partial F}{\partial \Sigma_{v\to w}}\right)_{(v,w)\in(\mathcal{V}\setminus\mathcal{V}^{(0)})\times(\mathcal{V}\setminus\mathcal{V}^{(0)})} \tag{39}$$

we obtain

$$\frac{\partial F}{\partial \Sigma} = \frac{1}{2}\Sigma^{-2} \circ (\epsilon\epsilon^T - \Sigma) \tag{40}$$

Proof. Let $v, w \in \mathcal{V} \setminus \mathcal{V}^{(0)}$.

$$\frac{\partial F}{\partial \Sigma_{v\to w}} = -\frac{1}{2}\left(\frac{\partial}{\partial \Sigma_{v\to w}} \ln\det\Sigma + \frac{\partial}{\partial \Sigma_{v\to w}} \epsilon^T\Sigma^{-1}\epsilon\right) \tag{41}$$

We obtain

$$\frac{\partial}{\partial \Sigma_{v\to w}} \ln\det\Sigma = \Sigma^{-1}_{w\to v} = \Sigma^{-1}_{v\to w} \tag{42}$$

3 Open Problem (Justification of (42)). Reconstruct the solution.

On the other hand,

$$\frac{\partial \epsilon^T\Sigma^{-1}\epsilon}{\partial \Sigma^{-1}} = \epsilon\epsilon^T \tag{43}$$

togther with

$$\frac{\partial \Sigma^{-1}}{\partial \Sigma} = -\Sigma^{-2} \tag{44}$$

yields

$$\frac{\partial}{\partial \Sigma_{v\to w}} \epsilon^T\Sigma^{-1}\epsilon = \frac{\partial}{\partial \Sigma^{-1}_{v\to w}} \epsilon^T\Sigma^{-1}\epsilon \frac{\partial \Sigma^{-1}_{v\to w}}{\partial \Sigma_{v\to w}} = -\varepsilon_v\varepsilon_w \left(\Sigma^{-2}\right)_{v\to w} \tag{45}$$

Combining (42,45) yields (40). □

2.16 (Learning Σ I) Gradient ascent is clearly not Hebbian if Σ is not diagonal, that is, if the error terms are correlated. Note that, we cannot obtain ϵ either. However, the direction of the gradient is given by $\epsilon\epsilon^T - \Sigma$ and at least does not make computation of the gradient necessary. [1] proposes a solution for a less general situation.

4 Open Problem (Hebbian Σ). How can we learn Σ in a Hebbian way without adding more PUs than necessary?

References

[1] Rafal Bogacz. "A tutorial on the free-energy framework for modelling perception and learning". In: *Journal of Mathematical Psychology* 76 (2017), pp. 198–211.

[2] A. P. Dempster, N. M. Laird, and D. B. Rubin. "Maximum Likelihood from Incomplete Data via the EM Algorithm". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 39.1 (1977), pp. 1–38.

[3] Karl Friston. "A theory of cortical responses". In: *Philosophical Transactions of the Royal Society B* 360 (2005), pp. 815–836. DOI: 10.1098/rstb.2005.1622.

[4] James C. R. Whittington and Rafal Bogacz. "An Approximation of the Error Backpropagation Algorithm in a Predictive Coding Network with Local Hebbian Synaptic Plasticity". In: *Neural Computation* 29 (2017), pp. 1229–1262. ISSN: 1530888X. DOI: 10.1162/NECO. arXiv: 1706.02451. URL: http://arxiv.org/abs/1706.02451.