Notation of Neural Networks

Samuel Lippl

April 22, 2018

Contents

While the choice of a specific notation is mathematically irrelevant, a notation that is

- concise (so that it is as easy as possible to write and read)
- consistent (so that there is no confusion) and
- clear (where every building block of the system is where one would expect it; this has a lot to do with consistency)

is obviously important. In this document, I propose a notation that attempts to suffice these criteria. If some tweak turns out to be superior, I will change the document.

Building Blocks As the elementary building blocks of a neural network the **Processing Units**/Nodes ($u, v, w, \dots$) and the **Unit Connections**/Edges ($c = (v, w) = v \to w$) do not need any subscripts – a simply symbol suffices where $v \to w$ is used both to reference (v, w) and to notate that $(v, w) \in \mathcal{C}$. These notations are also acceptable for sets of PUs/UCs.
The **preceding** (resp. **subsequent**) PUs are written as $\to v$ (resp. $v \to$).
The **weights** of a UC c are written as $\theta_c = \theta_{v \to w}$. We may also use sets of PUs: $\theta_V \equiv (\theta)_{v \in V}$. We will identify $v \to \equiv (v \to w)_{w \in v \to}$ where it will become clear from context whether we mean PUs or UCs. We will therefore use that notation $\theta_{v\to}, \theta_{\to v}$.
Generally, nodes are written in brackets, if we look at a **state** s, the state of a particular PU v if written as $s(v)$. Again, $s(V) \equiv (s(v))_{v \in V}$. We mostly use the letters s, i (for an input state) or o (for an output). States are generally used as a superscript.
General **network functions** are now written as N_θ^i where i is their input. They map to states, i. e. $N_\theta^i(v)$ denotes the state of PU v. If there is a set of output PUs v^O (normally $v^{(L)}$), we write $N_\theta^i(v) \equiv O_\theta^i(v)$. If we want to talk about the function that maps input to states (resp. real numbers) we write $N_\theta^{\cdot}$ (resp. $N_\theta^{\cdot}(v)$).
Unit Functions are the only context in which we use PUs as subscripts: $f_v, g_v, f_{\to v} \equiv (f_w)_{w \in (\to v)}, \dots$
We denote **learning functions** by writing the current weights θ in brackets: $L^i(\theta)$ where i is the input or, if it is a learning function that makes a difference between input and output, we may clarify that by $L^{i;o}(\theta)$. As the domain is the weights domain $L_c^{i;o}(\theta)$ references a newly updated weight.

Entire networks The notations above do not yet provide a concise notation for an entire network. Generally, we will write networks as $\mathcal{N}$ but specific networks may be written with fitting letters, for instance: $\mathcal{B}$ (for a backpropagation network) or $\mathcal{P}$ (for a predictive coding network). The only necessary parts of a network are PUs and UCs: $\mathcal{N} = (\mathcal{V}, \mathcal{C})$. We may also use a matrix notation which requires some preliminary work.
The goal is to denote both the weights and the network structure by a matrix θ where the rows and columns are indexed by the set of PUs $\mathcal{V}$. However, in order to be able to learn, the matrix would need to save information on which connections are able to be trained. This applies both to constantly weighted UCs and to PUs that are not connected and therefore have the weight 0. On the other hand, we would like to denote the network's structure even in situations where we have not obtained some of the weights yet. My proposed solution is the following:

0.1 Notation. We define

$$\mathbb{R}_{\varsigma,\chi} := (\mathbb{R} \times \{\varsigma, \chi\}) \cup \{\chi\} \tag{1}$$

where we denote for any $r \in \mathbb{R}$

$$r_\varsigma \equiv (1, \varsigma) \qquad r \equiv r_\chi \equiv (1, \chi) \tag{2}$$

We may omit 0_ς but not 0 and use $\chi_1, \ldots, \chi_k$ to denote different classes of UCs. As χ simply represents a parameter that is not specified, we will mostly simply write that parameter θ_c.

0.2 (Matrix Notation of Neural Networks) In the matrix notation, we specify neural networks by a weight matrix $\theta \in \mathbb{R}_{\varsigma,\chi}^{\mathcal{V}\times\mathcal{V}}$ where $\theta_{i,j}$ specifies a connection from i to j, i. e.:

$$W = \begin{pmatrix} \theta_{v_1 \to v_1} & \cdots & \theta v_1 \to v_n \\ \vdots & \ddots & \vdots \\ \theta_{v_n \to v_1} & \cdots & \theta v_n \to v_n \end{pmatrix} \tag{3}$$

Everything but 0_ς specifies a connection where the constant values denote constant edges while r_χ denotes a current value of a parameter and χ specifies that PUs are connected by a parameter with no current value.

0.3 Notation (Execution of a function). We define

$$f \rhd (x_i)_{i\in I} \equiv (f(x_i))_{i\in I} \equiv (x_i)_{i\in I} \lhd f \qquad (f_i)_{i\in I} \blacktriangleright (x_i)_{i\in I} \equiv (f_i(x_i))_{i\in I} \equiv (x_i)_{i\in I} \blacktriangleleft (f_i)_{i\in I} \tag{4}$$

0.4 Notation (Pairwise multiplication).

$$(x_i)_{i\in I} \circ (y_i)_{i\in I} \equiv (x_i y_i)_{i\in I} \tag{5}$$

0.5 (Step Function in matrix notation) We may now notate the step function as

$$T(s) := \theta f \rhd s \tag{6}$$

0.6 Notation (Recursive Definitions). We use $\downharpoonleft$ to make recursive definitions more concise. If we have a lower triagonal matrix (e. g. a feedforward network) $W \in \mathbb{R}^{n\times n}$, we may define a matrix

$$R := Wr \downharpoonleft \tag{7}$$

where $r \in \mathbb{R}^k$ and W must have at least k rows of zeros. Then, R is initialized in the first k rows by r and then recursively defined by W:

$$R_i := \begin{cases} r_i & i \leq k \\ W_{i\cdot} \begin{pmatrix} r_1 \\ \vdots \\ r_{i-1} \\ 0 \\ \vdots \\ 0 \end{pmatrix} & k < i \leq n \end{cases} \tag{8}$$

If W is an upper triagonal matrix, we may also write

$$R := Wr \upharpoonleft \tag{9}$$

with the corresponding definition.
I will also extend this notation to expressions like

$$Wf \rhd i \downharpoonleft \tag{10}$$

0.7 (10) describes the canonical state function where $i = \mathcal{V}^{(0)}$.