I

Notes on Supervised Learning

§ 1.1 Backgrounds

Let's denote the set $X_0 = \{x_0^k\}_{k=1,\dots,K}$ as the collection of K-numbers of abstract discrete data points (the subscript 0 means that it's an unprocessed 'raw' data). In the theory of supervised learning, every data set X_0 poses a 'class' for each data point to be associated with.

Definition 1: (Class) ***Classes*** *$\{\mathcal{C}_m\}$ of the data set X_0 is a partition of X_0, i.e.,*

$$\bigcup_m \mathcal{C}_m = X_0, \tag{1}$$

and every data point x^k belongs to each unique partition $\mathcal{C}_m$.

The number of partitioned subsets (i.e., the number of classes) will now denoted as n_c.

Let $\mathbb{P}^{n_c}$ be a probability space of dimension n_c, i.e., for any $\mathbf{p} \in \mathbb{P}^{n_c}$, its each index $(\mathbf{p})_i$, $i = 1, \dots, n_c$ is non-negative and $\sum_i (\mathbf{p})_i = 1$. Then we can label each x_0^k by defining the *representation* of classes in terms of a probability vector in $\mathbb{P}^{n_c}$, which is legally called as *the ground truth* of x_0^k.

Definition 2: (Ground Truth) *For $x_0^k \in \mathcal{C}_m$, the* ***ground truth*** *of data point x_0^k is a one-hot vector $\mathbf{y}^k \in \mathbb{P}^{n_c}$ where*

$$(\mathbf{y}^k)_j = 1 \quad \text{if} \quad j = m, \;\; \text{and else} \;\; 0. \tag{2}$$

For example, if $n_c = 4$ for given X_0 and $x_0^k \in \mathcal{C}_2$ for some k, then

$$\mathbf{y}^k = [0 \;\; 1 \;\; 0 \;\; 0]^\top. \tag{3}$$

We now define a mapping called *classifier*, which will be our main interest:

Definition 3: (Classifier) *A* ***classifier*** *$\mathbf{M}$ for the set X_0, say $\mathbf{M}(X_0)$, is a mapping*

$$\mathbf{M} : X_0 \to \mathbb{P}^{n_c}, \quad \text{i.e.,} \quad \mathbf{M}(x^k) \in \mathbb{P}^{n_c}. \tag{4}$$

Above definition naturally leads to such interpretation: m-th component of $\mathbf{M}(x_0^k)$ is *the probability that how much the classifier $\mathbf{M}$ regards that $x_0^k \in \mathcal{C}_m$*. This directly yields the final goal of supervised learning, which is to obtain an ideal map $\mathbf{M}$ per data set X_0 that accurately maps each x_0^k to its corresponding ground truth $\mathbf{y}_k$. To handle this concept mathematically, we introduce an vector-represented *embedding* of x_0^k as the following:

Definition 4: (Embedding) *For any bijective mapping $E : X_0 \to \mathbb{R}^{n_f}$, we call*

$$E(x_0^k) \in \mathbb{R}^{n_f} \tag{5}$$

*as the **embedding** of a data point x^k.*

The dimension n_f of the embedded data point is often called as the *number of features.*

For convenience, we will simply refer to $E(x_0^i)$ as $\mathbf{x}_0^i$ under the assumption that an appropriate embedding map E is given for X_0. Equipped with such vector embedding, the above notion of (4) in Definition 3 can be rewritten as follows:

$$\mathbf{M} : \mathbb{R}^{n_f} \to \mathbb{P}^{n_c}, \quad \text{i.e.,} \quad \mathbf{M}(\mathbf{x}_0^i \in \mathbb{R}^{n_f}) \in \mathbb{P}^{n_c}. \tag{6}$$

Here, we introduce an important concept of 'how inaccurate the classifier $\mathbf{M}$ is'. It is measured through the definition of following *loss* function:

Definition 5: (Loss) *Let $\mathbf{y}^k$ be the ground truth of each data point x_0^k. Then, the **loss** function* $\text{loss} : \mathbb{P}^{n_f} \times \mathbb{P}^{n_f} \to \mathbb{R}^+$ *for each x_0^k is defined as follows:*

$$\text{loss}(\mathbf{M}(\mathbf{x}^k); \mathbf{y}^k) \geq 0, \tag{7}$$

where its equality only holds when $\mathbf{M}(\mathbf{x}^k) = \mathbf{y}^k$.

Thus, writing $\mathbf{X}_0 = \{\mathbf{x}_0^k\}_{k=1,\ldots,K}$ as the whole embedded data set, the main goal can be simply reduced as follows: *Finding an optimal parameter family $\mathbf{W}$ for the map $\mathbf{M}(\mathbf{X}_0; \mathbf{W})$, that achieves*

$$\mathbf{W}^* = \text{argmin}_{\mathbf{W}} \, \text{Loss}(\mathbf{M}(\mathbf{X}_0; \mathbf{W}), \mathbf{Y}), \tag{8}$$

where function Loss denotes the *global* loss function across $\mathbf{M}(\mathbf{X}_0; \mathbf{W})$ and its collection of ground truths $\mathbf{Y}$, i.e., $\mathbf{Y} = \{\mathbf{y}^k\}_{k=1,\ldots,K}$ (from now, a simple notation $\mathbf{X}_{\text{out}}$ will be used to denote $\mathbf{M}(\mathbf{X}_0)$).

The solution of Eq. (8) can be found by searching for $\mathbf{W}^*$ satisfying the global parameter derivative equation

$$\frac{\partial}{\partial \mathbf{W}} \text{Loss}(\mathbf{X}_{\text{out}}; \mathbf{Y}) \bigg|_{\mathbf{W}=\mathbf{W}^*} = \mathbf{0}, \tag{9}$$

however, even if the structure of classifier dynamics becomes a little bit complicated, it will be almost impossible to acquire the analytic solution of above Eq. (9). But, however, from the next section we will see that only computing the derivative $\frac{\partial}{\partial \mathbf{W}} \text{Loss}(\mathbf{X}_{\text{out}}; \mathbf{Y})$ is quite analytically eligible, so we can perform steepest descent algorithm using such analytically given derivative, i.e.,

$$\mathbf{W}^{(i+1)} = \mathbf{W}^{(i)} + \alpha \frac{\partial}{\partial \mathbf{W}} \text{Loss}(\mathbf{X}_{\text{out}}; \mathbf{Y}) \bigg|_{\mathbf{W}=\mathbf{W}^{(i)}}. \tag{10}$$

Furthermore, since $\mathbf{Y}$ is globally fixed and only $\mathbf{X}_{\text{out}}$ depends on $\mathbf{W}$, observe that

$$\frac{\partial}{\partial \mathbf{W}} \text{Loss}(\mathbf{X}_{\text{out}}; \mathbf{Y}) = \frac{\partial}{\partial \mathbf{X}_{\text{out}}} \text{Loss}(\mathbf{X}_{\text{out}}; \mathbf{Y}) \frac{\partial}{\partial \mathbf{W}} \mathbf{X}_{\text{out}} \tag{11}$$

$$= \sum_{k=1}^{K} \underbrace{\frac{\partial}{\partial \mathbf{x}^k_{\text{out}}} \text{loss}(\mathbf{x}^{\text{k}}_{\text{out}}; \mathbf{y}^{\text{k}})}_{\text{(A)}} \underbrace{\frac{\partial}{\partial \mathbf{W}} \mathbf{x}^k_{\text{out}}}_{\text{(B)}}. \tag{12}$$

Thus we arrive at the following important insight:

Computing the global parameter derivative can be divided into two parts (A) and (B), where (A) only depends on the structure of loss criterion, and (B) depends on the structure of classifier $\mathbf{M}(\mathbf{x}^k_0; \mathbf{W}) = \mathbf{x}^k_{\text{out}}$.

In this sense, from now we will mainly focus on exploring and analyzing various classifier structures, aside from the materials related to loss evaluation criteria. For missed details, refer to § 1.4.

§ 1.2 Classifiers based on Feed-forward Dynamics

§ 1.2.1 Concepts of Feed-forward Machines

Here, we first give the most popular method of constructing such classifier map $\mathbf{M}$ for a given discrete data set X_0, through the structure of *feed-forward* dynamics.

A feed-forward machine is structurally defined as follows:

Definition 6: (Feed-forward Machine) *A feed-forward machine is a collection of* ***layers****, say,* $\{\mathcal{L}_l(\mathbf{x}; \mathbf{W}_l)\}_{l=0,1,\ldots}$ *that performs a sequential transformation of data embedding* $\mathbf{x}^k_0$ *depending on its each layer parameters* $\mathbf{W}_l$.

More precisely, an initial embedding $\mathbf{x}^k_0$ is carried into the first layer as $\mathcal{L}_0(\mathbf{x}^k_0; \mathbf{W}_0)$, and the layer $\mathcal{L}_0$ transforms the input $\mathbf{x}^k_0$ into $\mathbf{x}^k_1$. If there are $L+1$ numbers of layers, one can rewrite as

$$\mathbf{x}^k_{l+1} = \mathcal{L}_l(\mathbf{x}^k_l; \mathbf{W}_l), \quad l = 0, 1, \ldots, L \tag{13}$$

(see Figure 1-1), and such feed-forward procedure $\mathbf{x}^k_0 \to \mathbf{x}^k_1 \to \cdots$ is called *message passing.*

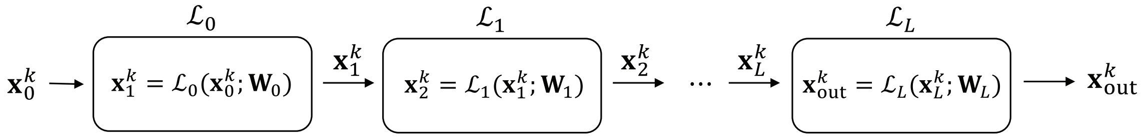
$\mathcal{L}_0$
$\mathcal{L}_1$
$\mathcal{L}_L$
$\mathbf{x}^k_0$
$\mathbf{x}^k_1 = \mathcal{L}_0(\mathbf{x}^k_0; \mathbf{W}_0)$
$\mathbf{x}^k_1$
$\mathbf{x}^k_2 = \mathcal{L}_1(\mathbf{x}^k_1; \mathbf{W}_1)$
$\mathbf{x}^k_2$
$\cdots$
$\mathbf{x}^k_L$
$\mathbf{x}^k_{\text{out}} = \mathcal{L}_L(\mathbf{x}^k_L; \mathbf{W}_L)$
$\mathbf{x}^k_{\text{out}}$

Figure 1-1: An illustrative diagram for feed-forward message passing.

Here, if the last transformed output $\mathbf{x}^k_{L+1}$ is in the space of $\mathbb{P}^{n_c}$ (details for this, see the next section), the collection of feed-forward layers $\{\mathcal{L}_l\}_{l=0,1,\ldots,L}$ itself can solely act as a classifier mapping $\mathbf{M}$ for the embedded data set $\mathbf{X}_0$. Thus, if we denote $\mathbf{x}^k_{L+1}$ as $\mathbf{x}^k_{\text{out}}$, then one can use Definition 5 to measure $\text{loss}(\mathbf{x}^k_{\text{out}}, \mathbf{y}^k)$ for the data point x^k_0, predicted by the feed-forward machine $\{\mathcal{L}_l\}_{l=0,1,\ldots,L}$.

Now, the only thing left is to find the optimal parameter family $\mathbf{W}^* = \{\mathbf{W}_l^*\}_{l=0,1,\ldots,L}$ in the sense of Eq. (8) and (9), i.e.,

$$\mathbf{W}^* = \mathrm{argmin}_{\mathbf{W}} \ \mathrm{Loss}(\mathbf{X}_{\mathrm{out}}; \mathbf{Y}). \tag{14}$$

Without concerning the structure of global loss criterion, Eq. (12) tells us that the only thing to be mattered is the derivative of the classifier dynamics, i.e., $\frac{\partial}{\partial \mathbf{W}} \mathbf{x}_{\mathrm{out}}^k$, and we first give the direct result:

Theorem 7: *For any feed-forward dynamics by $L+1$ layers,*

$$\begin{aligned}\frac{\partial}{\partial \mathbf{W}} \mathbf{x}_{\mathrm{out}} = & \frac{\partial \mathcal{L}_L(\mathbf{x}_L; \mathbf{W}_L)}{\partial \mathbf{W}_L} \frac{\partial \mathbf{W}_L}{\partial \mathbf{W}} \\ & + \sum_{l=0}^{L-1} \left(\frac{\partial \mathcal{L}_L(\mathbf{x}_L; \mathbf{W}_L)}{\partial \mathbf{x}_L} \frac{\partial \mathcal{L}_{L-1}(\mathbf{x}_{L-1}; \mathbf{W}_{L-1})}{\partial \mathbf{x}_{L-1}} \cdots \frac{\partial \mathcal{L}_l(\mathbf{x}_l; \mathbf{W}_l)}{\partial \mathbf{W}_l} \right) \frac{\partial \mathbf{W}_l}{\partial \mathbf{W}},\end{aligned} \tag{15}$$

where superscript k of each feed-forwarded data point $\mathbf{x}_l^k$ has been omitted.

Proof. From $\mathbf{x}_{\mathrm{out}} = \mathcal{L}_L(\mathbf{x}_L; \mathbf{W}_L)$, note that $\mathbf{x}_L$ is independent of $\mathbf{W}_L$ (so only dependent on $\mathbf{W}_0, \ldots, \mathbf{W}_{L-1}$). Thus introducing a new notation

$$\mathbf{W}_{0,1,\ldots,l} = \{\mathbf{W}_0, \mathbf{W}_1, \ldots, \mathbf{W}_l\}, \tag{16}$$

one can more specifically write the transform rule by $\mathcal{L}_L$ regarding the dependencies on each layer parameters $\mathbf{W}_l$ as $\mathcal{L}_L(\mathbf{x}_L(\mathbf{W}_{0,\ldots,L-1}); \mathbf{W}_L)$. Then by chain rule, we observe

$$\frac{\partial}{\partial \mathbf{W}} \mathbf{x}_{\mathrm{out}} = \frac{\partial}{\partial \mathbf{W}} \mathcal{L}_L(\mathbf{x}_L(\mathbf{W}_{0,\ldots,L-1}); \mathbf{W}_L) \tag{17}$$

$$\begin{aligned} = & \frac{\partial \mathcal{L}_L(\mathbf{x}_L(\mathbf{W}_{0,\ldots,L-1}; \mathbf{W}_L))}{\partial \mathbf{x}_L(\mathbf{W}_{0,\ldots,L-1})} \underbrace{\frac{\partial \mathbf{x}_L(\mathbf{W}_{0,\ldots,L-1})}{\partial \mathbf{W}_{0,\ldots,L-1}}}_{(*)} \frac{\partial \mathbf{W}_{0,\ldots,L-1}}{\partial \mathbf{W}} \\ & + \frac{\partial \mathcal{L}_L(\mathbf{x}_L(\mathbf{W}_{0,\ldots,L-1}))}{\partial \mathbf{W}_L} \frac{\partial \mathbf{W}_L}{\partial \mathbf{W}}.\end{aligned} \tag{18}$$

Note that each multiplication of derivatives is actually a tensor contraction operation among corresponding co- and contra-variant dimensions. Also, note that the term $(*)$ is exactly same with the global derivative for only L-numbers of layers, i.e., layers except the last layer $\mathcal{L}_L$. Therefore

$$\begin{aligned}(*) = & \frac{\partial \mathcal{L}_{L-1}(\mathbf{x}_{L-1}(\mathbf{W}_{0,\ldots,L-2}; \mathbf{W}_{L-2}))}{\partial \mathbf{x}_{L-1}(\mathbf{W}_{0,\ldots,L-2})} \underbrace{\frac{\partial \mathbf{x}_{L-1}(\mathbf{W}_{0,\ldots,L-2})}{\partial \mathbf{W}_{0,\ldots,L-2}}}_{(**)} \frac{\partial \mathbf{W}_{0,\ldots,L-2}}{\partial \mathbf{W}_{0,\ldots,L-1}} \\ & + \frac{\partial \mathcal{L}_{L-1}(\mathbf{x}_{L-1}(\mathbf{W}_{0,\ldots,L-2}))}{\partial \mathbf{W}_{L-1}} \frac{\partial \mathbf{W}_{L-1}}{\partial \mathbf{W}_{0,\ldots,L-1}},\end{aligned} \tag{19}$$

(also note for the term $(**)$) and putting this term into Eq. (18) and omitting the redundant dependency notations for clearance, we get

$$\frac{\partial}{\partial \mathbf{W}} \mathbf{x}_{\mathrm{out}} = \frac{\partial \mathcal{L}_L}{\partial \mathbf{x}_L} \frac{\partial \mathcal{L}_{L-1}}{\partial \mathbf{x}_{L-1}} \underbrace{\frac{\partial \mathbf{x}_{L-1}}{\partial \mathbf{W}_{0,\ldots,L-2}}}_{(**)} \frac{\partial \mathbf{W}_{0,\ldots,L-2}}{\partial \mathbf{W}} \tag{20}$$

$$+\frac{\partial\mathcal{L}_L}{\partial\mathbf{x}_L}\frac{\partial\mathcal{L}_{L-1}}{\partial\mathbf{W}_{L-1}}\frac{\partial\mathbf{W}_{L-1}}{\partial\mathbf{W}}+\frac{\partial\mathcal{L}_L}{\partial\mathbf{W}_L}\frac{\partial\mathbf{W}_L}{\partial\mathbf{W}}, \tag{21}$$

where the tensor relations

$$\frac{\partial\mathbf{W}_{0,\dots,L-2}}{\partial\mathbf{W}_{0,\dots,L-1}}\frac{\partial\mathbf{W}_{0,\dots,L-1}}{\partial\mathbf{W}}=\frac{\partial\mathbf{W}_{0,\dots,L-2}}{\partial\mathbf{W}}\quad\text{and} \tag{22}$$

$$\frac{\partial\mathbf{W}_{L-1}}{\partial\mathbf{W}_{0,\dots,L-1}}\frac{\partial\mathbf{W}_{0,\dots,L-1}}{\partial\mathbf{W}}=\frac{\partial\mathbf{W}_{L-1}}{\partial\mathbf{W}} \tag{23}$$

has been used.

Repeating this procedure recurrently, then we finally get

$$\begin{aligned}\frac{\partial}{\partial\mathbf{W}}\mathbf{x}_{\text{out}}= &\ \frac{\partial\mathcal{L}_L}{\partial\mathbf{x}_L}\frac{\partial\mathcal{L}_{L-1}}{\partial\mathbf{x}_{L-1}}\cdots\frac{\partial\mathcal{L}_1}{\partial\mathbf{x}_1}\frac{\partial\mathbf{x}_1(=\mathcal{L}_0)}{\partial\mathbf{W}_0}\frac{\partial\mathbf{W}_0}{\partial\mathbf{W}} && (24)\\ &+\frac{\partial\mathcal{L}_L}{\partial\mathbf{x}_L}\frac{\partial\mathcal{L}_{L-1}}{\partial\mathbf{x}_{L-1}}\cdots\frac{\partial\mathbf{x}_2(=\mathcal{L}_1)}{\partial\mathbf{W}_1}\frac{\partial\mathbf{W}_1}{\partial\mathbf{W}} && (25)\\ &\ \vdots\\ &+\frac{\partial\mathcal{L}_L}{\partial\mathbf{x}_L}\frac{\partial\mathcal{L}_{L-1}}{\partial\mathbf{W}_{L-1}}\frac{\partial\mathbf{W}_{L-1}}{\partial\mathbf{W}} && (26)\\ &+\frac{\partial\mathcal{L}_L}{\partial\mathbf{W}_L}\frac{\partial\mathbf{W}_L}{\partial\mathbf{W}}, && (27)\end{aligned}$$

which is equivalent to our desired result, Eq. (15) in Theorem 7. ■

Remark.

a. Since the expressions $\frac{\partial\mathbf{W}_l}{\partial\mathbf{W}}$ in Eq. (24) to (27) only acts as assigning the formerly multiplied derivative tensors to the basis space of $\mathbf{W}_l$ corresponding in global parameter family $\mathbf{W}$, one can alternatively read that

$$\boxed{\begin{aligned}\frac{\partial}{\partial\mathbf{W}_L}\mathbf{x}_{\text{out}}&=\frac{\partial\mathcal{L}_L(\mathbf{x}_L;\mathbf{W}_L)}{\partial\mathbf{W}_L}, && (28)\\ \frac{\partial}{\partial\mathbf{W}_{L-1}}\mathbf{x}_{\text{out}}&=\frac{\partial\mathcal{L}_L(\mathbf{x}_L;\mathbf{W}_L)}{\partial\mathbf{x}_L}\frac{\partial\mathcal{L}_{L-1}(\mathbf{x}_{L-1};\mathbf{W}_{L-1})}{\partial\mathbf{W}_{L-1}}, && (29)\\ &\ \vdots\\ \frac{\partial}{\partial\mathbf{W}_0}\mathbf{x}_{\text{out}}&=\frac{\partial\mathcal{L}_L(\mathbf{x}_L;\mathbf{W}_L)}{\partial\mathbf{x}_L}\frac{\partial\mathcal{L}_{L-1}(\mathbf{x}_{L-1};\mathbf{W}_{L-1})}{\partial\mathbf{x}_{L-1}}\cdots\frac{\partial\mathcal{L}_0(\mathbf{x}_0;\mathbf{W}_0)}{\partial\mathbf{W}_0}. && (30)\end{aligned}}$$

This layer-wise chain structured property of derivatives is widely referred to as the *back-propagation rule*. This is the key mechanism for performing steeped parameter descent in feed-forward dynamics.

b. The virtue of above result is that the global parameter derivative can be evaluated only with the sequential information of local derivatives of each layers through back-propagation rule, so that only requiring $\frac{\partial\mathcal{L}_l}{\partial\mathbf{x}_l}$ or $\frac{\partial\mathcal{L}_l}{\partial\mathbf{W}_l}$. □

Example 8: Let's consider the following very simple 3-layered feed-forward dynamics (regardless of the condition that $\mathbf{x}_{\text{out}} \in \mathbb{P}^{n_c}$): For $x_0 \in \mathbb{R}$,

$$x_1 = \mathcal{L}_0(x_0; w_0) = w_0 x_0, \tag{31}$$

$$x_2 = \mathcal{L}_1(x_1; w_1) = w_1^3 x_1, \tag{32}$$

$$x_{\text{out}} = \mathcal{L}_2(x_2; w_2) = w_2^2 x_2. \tag{33}$$

Then, one can straightforwardly read that $x_{\text{out}} = w_2^2 w_1^3 w_0 x_0$, so setting $\mathbf{W} = [w_0 \; w_1 \; w_2]^\top$,

$$\frac{\partial x_{\text{out}}}{\partial \mathbf{W}} = \left[\frac{\partial}{\partial w_0} \; \frac{\partial}{\partial w_1} \; \frac{\partial}{\partial w_2}\right] w_2^2 w_1^3 w_0 x_0 \tag{34}$$

$$= \left[w_2^2 w_1^3 x_0 \;\; 3w_2^2 w_1^2 w_0 x_0 \;\; 2w_2^2 w_1^3 w_0 x_0\right]. \tag{35}$$

This result can be also derived using the back-propagation rule (Eq. (28) to (30)). Observe

$$\frac{\partial x_{\text{out}}}{\partial w_2} = \frac{\partial \mathcal{L}_2(x_2; w_2)}{\partial w_2} = 2w_2^2 x_2, \tag{36}$$

$$\frac{\partial x_{\text{out}}}{\partial w_1} = \frac{\partial \mathcal{L}_2(x_2; w_2)}{\partial x_2} \frac{\partial \mathcal{L}_1(x_1; w_1)}{\partial w_1} = w_2^2 \cdot 3w_1^2 x_1, \tag{37}$$

$$\frac{\partial x_{\text{out}}}{\partial w_0} = \frac{\partial \mathcal{L}_2(x_2; w_2)}{\partial x_2} \frac{\partial \mathcal{L}_1(x_1; w_1)}{\partial x_1} \frac{\partial \mathcal{L}_0(x_0; w_1)}{\partial w_0} = w_2^2 \cdot w_1^3 \cdot x_0, \tag{38}$$

and replacing x_2 with $w_1^3 x_1 = w_1^3 w_0 x_0$ and x_1 with $w_0 x_0$, we get Eq. (35).

§ 1.2.2 Multi-Layer Perceptrons

Multi-Layer Perceptrons(MLP) are the most typical form of feed-forward machines. A *perceptron* layer performs linear transformation to the input, and then applies element-wise non-linear behavior (called *activation*) in order to pass the output to its next layer. More precisely, a perceptron layer $\mathcal{L}_l$ processes its given input $\mathbf{x}_l$ as

$$\mathbf{x}_{l+1} = \mathcal{L}_l(\mathbf{x}_l; \mathbf{W}_l) = \mathbf{\Phi}_l(\mathbf{W}_l \mathbf{x}_l), \tag{39}$$

thus its parameter $\mathbf{W}_l$ is analogously a matrix in $\mathbb{R}^{\dim(\mathbf{x}_{l+1}) \times \dim(\mathbf{x}_l)}$.

Now suppose that every layers *except the final one* $\mathcal{L}_L$ shares the same activation functional $\mathbf{\Phi} : \mathbb{R}^{\dim(\mathbf{x}_{l+1})} \to \mathbb{R}^{\dim(\mathbf{x}_{l+1})}$, with a typical element-wise sigmoid, i.e., $\mathbf{\Phi} = [\phi, \;\; \phi, \;\; \cdots \;\; \phi]$, being

$$(\mathbf{\Phi}(\mathbf{x}_l))_i = \phi((\mathbf{x}_l)_i) = \frac{1}{1 - \mathrm{e}^{(\mathbf{x}_l)_i}}. \tag{40}$$

The final layer's activation $\mathbf{\Phi}_L$ is conventionally given as a *softmax* function (say, $\mathbf{\Psi}$) in order to parse the final output into probability space $\mathbb{P}^{n_c}$, where

$$(\mathbf{\Psi}(\mathbf{x}_l))_i = \frac{\mathrm{e}^{(\mathbf{x}_l)_i}}{\sum_{m=1}^{n_c} \mathrm{e}^{(\mathbf{x}_l)_m}}. \tag{41}$$

Let us first only consider the layers except the last layer, i.e., $l = 0, 1, \ldots, L-1$. Then for each l, the value of $\frac{\partial \mathcal{L}_l}{\partial \mathbf{x}_l}$ and $\frac{\partial \mathcal{L}_l}{\partial \mathbf{W}_l}$ can be simply given as stated in the following theorem:

Theorem 9: *For any MLP network with $L+1$ numbers of layers, the local derivatives except the last layer can be computed with only using $\mathbf{W}_l$ and $\mathbf{x}_l$, that is, the values acquired during feed-forward message passing. Specifically, for $l = 0, 1, \ldots, L-1$,*

$$\frac{\partial \mathcal{L}_l(\mathbf{x}_l; \mathbf{W}_l)}{\partial \mathbf{x}_l} = \boldsymbol{\eta} \mathbf{W}_l, \quad \text{and} \quad \frac{\partial \mathcal{L}_l(\mathbf{x}_l; \mathbf{W}_l)}{\partial \mathbf{W}_l} = \boldsymbol{\eta} \otimes \mathbf{x}_l, \tag{42}$$

where $\boldsymbol{\eta} = \mathrm{Diag}\{\boldsymbol{\Phi}(\mathbf{W}_l \mathbf{x}_l) \odot (\mathbf{1} - \boldsymbol{\Phi}(\mathbf{W}_l \mathbf{x}_l))\} \in \mathbb{R}^{\dim(\mathbf{x}_{l+1}) \times \dim(\mathbf{x}_{l+1})}$, and $\odot$ denotes the element-wise Hadamard product.

Proof. Let us omit the layer index subscript l in order to avoid confusion with covariant dimension index subscripts. First, for $\frac{\partial}{\partial \mathbf{x}} \mathcal{L}(\mathbf{x}; \mathbf{W})$, setting $\mathbf{h} = \mathbf{W}\mathbf{x}$ and using Einstein's summation convention on tensor contraction expressions, we compute

$$\begin{aligned}
\left(\frac{\partial \mathcal{L}(\mathbf{x}; \mathbf{W})}{\partial \mathbf{x}} \right)^i_j &= \left(\frac{\partial \boldsymbol{\Phi}(\mathbf{h})}{\partial \mathbf{h}} \right)^i_k \left(\frac{\partial \mathbf{h}}{\partial \mathbf{x}} \right)^k_j &\quad (43)\\
&= \frac{\partial \phi(h^i)}{\partial h_k} \frac{\partial W^k_n x^n}{\partial x_j} &\quad (44)\\
&= \phi'(h^i) \delta^i_k \cdot W^k_n \delta^n_j &\quad (45)\\
&= \phi'(h^i) W^i_j \equiv \mathrm{Diag}\{\boldsymbol{\Phi}'(\mathbf{W}\mathbf{x})\} \mathbf{W}, &\quad (46)
\end{aligned}$$

where from Eq. (45) to (46) the fact that $\delta^i_k \delta^n_j W^k_n = W^i_j$, and a slight abuse of notation $\boldsymbol{\Phi}'$ indicating element-wise derivative, i.e., $\boldsymbol{\Phi}' = [\phi' \ \phi' \ \cdots \ \phi']$ has been used.

Next, for $\frac{\partial}{\partial \mathbf{W}} \mathcal{L}(\mathbf{x}; \mathbf{W})$, we compute

$$\begin{aligned}
\left(\frac{\partial \mathcal{L}(\mathbf{x}; \mathbf{W})}{\partial \mathbf{W}} \right)^i_{jk} &= \left(\frac{\partial \boldsymbol{\Phi}(\mathbf{h})}{\partial \mathbf{h}} \right)^i_l \left(\frac{\partial \mathbf{h}}{\partial \mathbf{W}} \right)^l_{jk} &\quad (47)\\
&= \frac{\partial \phi(h^i)}{\partial h_l} \frac{\partial W^l_n x^n}{\partial W_{jk}} &\quad (48)\\
&= \phi'(h^i) \delta^i_l \cdot \delta^l_j \delta_{nk} x^n &\quad (49)\\
&= \phi'(h^i) \delta^i_j x_k \equiv \mathrm{Diag}\{\boldsymbol{\Phi}'(\mathbf{W}\mathbf{x})\} \otimes \mathbf{x}. &\quad (50)
\end{aligned}$$

Further, using the well known fact that $\phi'(x) = \phi(x)(1 - \phi(x))$, we have $\mathrm{Diag}\{\boldsymbol{\Phi}'(\mathbf{W}\mathbf{x})\} = \mathrm{Diag}\{\boldsymbol{\Phi}(\mathbf{W}_l \mathbf{x}_l) \odot (\mathbf{1} - \boldsymbol{\Phi}(\mathbf{W}_l \mathbf{x}_l))\}$, and finally attaching the omitted l layer index notations, we get the desired result. ■

Remark. Note that for the last layer, the thing only changes is the derivative information of softmax $\boldsymbol{\Psi}$ stored in $\boldsymbol{\eta}$ instead of sigmoid $\boldsymbol{\Phi}$. Thus one only needs to newly compute $\frac{\partial}{\partial \mathbf{h}} \boldsymbol{\Psi}(\mathbf{h})$. □

§ 1.2.3 Convolutional Layers

§ 1.2.4 Graph Neural Networks

§ 1.3 Classifiers based on Recurrent Dynamics

§ 1.4 Criteria for Loss Computations