3D Spatially Embedded Networks: Verifying Coordinate Backpropagation in a Feed Forward MLP

Pascal Tohouri

14 July 2025

Abstract

The purpose of this research note is to verify coordinate backpropagation for a spatially embedded feed-forward MLP. The author applies the chain rule to obtain loss-coordinate derivatives for a L-layer, N_l-width MLP embedded in 3D Euclidean Space. Mean-squared error loss is used.

1 Setup

In a prior note, the author derived the gradient descent update via backpropagation on a single spatially embedded neuron. In a dense multilayer perceptron, each layer contains multiple neurons that propagate information in parallel.

2 Example: Feed Forward MLP

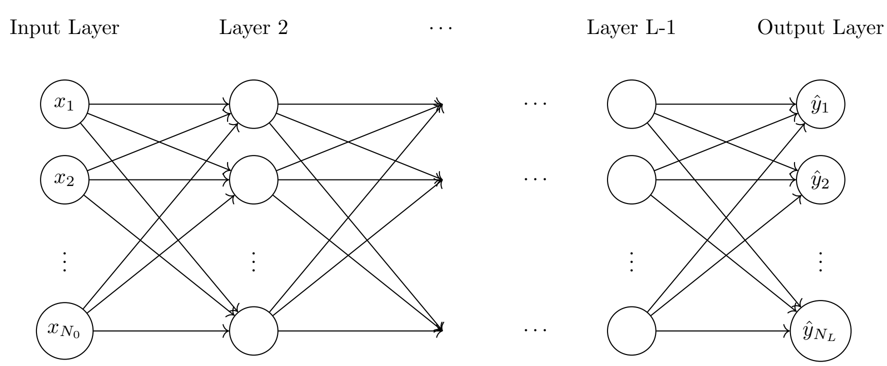

Figure 1: Dense MLP with L Layers and N Neurons per Layer

The neurons weight and sum incoming edge values, applying a differentiable activation:

$$h_{n_l}^{(l)} = \sum_{n_{l-1}} w_{n_l,n_{l-1}} h_{n_{l-1}}^{(l)} \tag{1}$$

$$a_{n_l}^{(l)} = f(h_{n_l}^{(l)}). \tag{2}$$

Activation values propagate across outgoing edges to the next layer. Once spatially embedded, the n-th node in layer l has 3D coordinates $\mathbf{z}_{n_l}^{(l)} = [z_1, z_2, z_3]$.

2.1 Parameter conditioning

Edge-weights depend on Euclidean distances between connecting nodes:

$$w^{(l)}_{n_l,n_{l-1}} = g(||\mathbf{z}^{(l)}_{n_l} - \mathbf{z}^{(l-1)}_{n_{l-1}}||). \tag{3}$$

2.2 Training

For loss $\mathcal{L}$, loss-weight derivatives are calculable via the chain rule:

$$\frac{\partial \mathcal{L}}{\partial w_{n_l,n_{l-1}}} = \delta^{(l)}_{n_l} \cdot h^{(l-1)}_{n_{l-1}} \tag{4}$$

Obtain loss-coordinate derivatives by chaining weight-coordinate derivatives:

$$\frac{\partial \mathcal{L}}{\partial z_{n_l,d}} = \sum_{n_{l-1}} \frac{\partial \mathcal{L}}{\partial w_{n_l,n_{l-1}}} \cdot \frac{\partial w_{n_l,n_{l-1}}}{\partial z_{n_l,d}} + \sum_{n_{l+1}} \frac{\partial \mathcal{L}}{\partial w_{n_{l+1},n_l}} \frac{\partial w_{n_{l+1},n_l}}{\partial z_{n_l,d}}, \tag{5}$$

where:

$$\frac{\partial w_{n_l,n_{l-1}}}{\partial z_{n_l}} = g'\big(\|\mathbf{z}_{n_l} - \mathbf{z}_{n_{l-1}}\|\big) \frac{z_{n_l} - z_{n_{l-1}}}{\|\mathbf{z}_{n_l} - \mathbf{z}_{n_{l-1}}\|} \quad \text{(Backward)} \tag{6}$$

$$\frac{\partial w_{n_{l+1},n_l}}{\partial z_{n_l,d}} = g'\big(\|\mathbf{z}_{n_{l+1}} - \mathbf{z}_{n_l}\|\big) \frac{z_{n_l,d} - z_{n_{l+1},d}}{\|\mathbf{z}_{n_{l+1}} - \mathbf{z}_{n_l}\|} \quad \text{(Forward)} \tag{7}$$

In full, the total loss-coordinate gradient is:

$$\frac{\partial \mathcal{L}}{\partial z_{n_l,d}} = \sum_{n_{l-1}} \Big[\delta^{(l)}_{n_l} h^{(l-1)}_{n_{l-1}} \cdot g'_{n_l,n_{l-1}}\Big] \frac{z_{n_l,d} - z_{n_{l-1},d}}{\|\mathbf{z}_{n_l} - \mathbf{z}_{n_{l-1}}\|} \quad + \quad \sum_{n_{l+1}} \Big[\delta^{(l+1)}_{n_{l+1}} h^{(l)}_{n_l} \cdot g'_{n_{l+1},n_l}\Big] \frac{z_{n_l,d} - z_{n_{l+1},d}}{\|\mathbf{z}_{n_{l+1}} - \mathbf{z}_{n_l}\|}. \tag{8}$$

Given the full coordinate gradient tensor for all layers l, nodes n_l, and spatial dimensions d:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{Z}} \in \ \mathbb{R}^{L \times N_l \times D}, \tag{9}$$

apply the gradient descent update step with learning rate η:

$$z^{(l)}_{n_l,d} \ \leftarrow \ z^{(l)}_{n_l,d} \ - \ \eta \, \frac{\partial \mathcal{L}}{\partial z^{(l)}_{n_l,d}}. \tag{10}$$

Equivalently, using tensor notation, the update rule is succinctly expressed as:

$$\mathbf{Z} \ \leftarrow \ \mathbf{Z} \ - \ \eta \, \frac{\partial \mathcal{L}}{\partial \mathbf{Z}}. \tag{11}$$

3 Conclusion

This research note presents the derivation of the gradient descent update condition for a spatially embedded feed-forward network with parameter conditioning. This method is generally applicable to feed-forward MLPs, but the note offers no empirical demonstration of the method's efficacy. Further work will empirically evaluate the method's efficiency, stability, and convergence.

A Notation

A.1 Function Objects

Neurons process input-output information:

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & \dots & x_{1,N_1} \\ \vdots & \ddots & \vdots \\ x_{N,1} & \dots & x_{N,N_1} \end{bmatrix} \tag{12}$$

$$\mathbf{Y} = \begin{bmatrix} y_{1,1} & \dots & y_{1,N_L} \\ \vdots & \ddots & \vdots \\ y_{N,1} & \dots & y_{N,N_L} \end{bmatrix} \tag{13}$$

where:

- n is the n-th data point $n \in \{1, \dots, N\}$
- n_1 is the n-th input feature (1st layer) $n_1 \in \{1, \dots, N_1\}$
- n_L is the n-th output feature (L-th layer) $n_L \in \{1, \dots, N_L\}$

Internal functions produce hidden states and activations:

$$\mathbf{h}^{(l)} = \begin{bmatrix} h_1 \\ \vdots \\ h_{N_l} \end{bmatrix}, \quad \mathbf{a}^{(l)} = \begin{bmatrix} a_1 \\ \vdots \\ a_{N_l} \end{bmatrix} \tag{14}$$

where:

- n_l is the n-th internal node in the l-th layer $n_l \in \{1, \dots, N_l\}$

A.2 Spatial Objects

Distinguish function objects from spatial coordinates. Here is a 'slice' of the network at the l-th layer:

$$\mathbf{Z}^{(l)} = \begin{bmatrix} z^{(l)}_{1,1} & \dots & z^{(l)}_{1,D} \\ \vdots & \ddots & \vdots \\ z^{(l)}_{N_l,1} & \dots & z^{(l)}_{N_l,D} \end{bmatrix}_{N_l \times D} \tag{15}$$

where:

- $z^{(l)}_{n_l,d}$ is the n_l-th node's d-th dimension coordinate $d \in \{1, \dots, D\}$, for the l-th layer $l \in \{1, \dots, \mathrm{L}\}$