1.1 Multilayer Perceptrons

1.1.1Linear to Nonlinear

1. Input

To construct a Multi-Layer Perceptron, We begin by characterizing the input structure through matrix

$$\mathbf{X} \in \mathbb{R}^{n \times d}$$

, representing a collection of n examples with d-dimensional features.

In the context of dynamical systems, n corresponds to the number of system variables while d represents the temporal evolution. The complete structure of the input transformation can be explicitly written as:

$$\mathbf{X} = [X_1, X_2, \cdots, X_n] = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{d,1} & x_{d,2} & \cdots & x_{d,n} \end{pmatrix}$$

2. Hidden

We consider a network architecture with a single hidden layer containing h neurons. The hidden layer transformation can be characterized by matrix

$$\mathbf{H} \in \mathbb{R}^{n \times h}$$

, where the dimensionality naturally emerges from processing n input examples through h hidden nodes. The explicit form of this transformation matrix is given by:

$$\mathbf{H} = \begin{pmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,h} \\ h_{2,1} & h_{2,2} & \cdots & h_{2,h} \\ \vdots & \vdots & \ddots & \vdots \\ h_{n,1} & h_{n,2} & \cdots & h_{d,h} \end{pmatrix}$$

3. Weights

Consider a fully connected network architecture where each layer is coupled to its adjacent layers through weight matrices and bias terms.

3.1 Input weight

The input-to-hidden layer coupling is characterized by weight matrix

$$\mathbf{W}^{(1)} \in \mathbb{R}^{d \times h}$$

. This dimensionality emerges naturally from the transformation of d-dimensional input features to h hidden neurons for each example in the ensemble.

3.2 input bias

To introduce additional degrees of freedom in the input transformation, we incorporate bias term

$$\mathbf{b}^{(1)} \in \mathbb{R}^{1\times h}$$

, where the dimensionality corresponds to the number of hidden neurons h.

3.3 output weight

The hidden-to-output layer transformation is governed by weight matrix

$$\mathbf{W}^{(2)} \in \mathbb{R}^{h\times q}$$

, facilitating the mapping from h hidden neurons to q output neurons.

3.4 output bias

Similarly, the output transformation includes bias term

$$\mathbf{b}^{(2)} \in \mathbb{R}^{1\times q}$$

, where the dimensionality aligns with the output layer structure.

Output

For an ensemble of n examples, the network output maintains dimensionality $n \times q$, preserving batch size while mapping to q-dimensional feature space. We now examine the temporal sampling methodology for dynamical systems, specifically applied to the Lorenz System.

Consider a phase space trajectory where the system state is characterized by variables (x_i, y_i, z_i) at discrete time points $i \in 1, 2, \cdots, T$. The sampling procedure involves two key parameters:

- Segment length d: number of consecutive temporal measurements
- Ensemble size n: number of distinct segments in the batch

To demonstrate the sampling methodology, we analyze two distinct configurations with parameters $n = 10$ and $d = 5$:

Configuration 1: Non-overlapping Sampling

The first sampling strategy employs disjoint segments:

$$E_1 = \{(x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3), (x_4, y_4, z_4), (x_5, y_5, z_5)\}$$

$$E_2 = \{(x_6, y_6, z_6), (x_7, y_7, z_7), (x_8, y_8, z_8), (x_9, y_9, z_9), (x_{10}, y_{10}, z_{10})\}$$
$$\vdots$$
$$E_{10} = \{(x_{46}, y_{46}, z_{46}), (x_{47}, y_{47}, z_{47}), (x_{48}, y_{48}, z_{48}), (x_{49}, y_{49}, z_{49}), (x_{50}, y_{50}, z_{50})\}$$

Here, segments E_n are constructed with stride equal to segment length d, ensuring temporal independence between successive segments.

Configuration 2: Unit-Stride Sampling

$$E_1 = \{(x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3), (x_4, y_4, z_4), (x_5, y_5, z_5)\}$$

$$E_2 = \{(x_2, y_2, z_2), (x_3, y_3, z_3), (x_4, y_4, z_4), (x_5, y_5, z_5), (x_6, y_6, z_6)\}$$
$$\vdots$$
$$E_{10} = \{(x_{10}, y_{10}, z_{10}), (x_{11}, y_{11}, z_{11}), (x_{12}, y_{12}, z_{12}), (x_{13}, y_{13}, z_{13}), (x_{14}, y_{14}, z_{14})\}$$

This sampling methodology maintains segment length d while implementing unit temporal displacement between successive segments. The stride parameter can be generalized to any integer value in $[1, d]$, providing flexible control over temporal sampling density.

Physical Interpretation of Sampling Strategy

The employed sampling methodology serves two fundamental purposes in capturing the system dynamics:

1. Batch Processing Architecture: The simultaneous processing of n distinct segments enables comprehensive sampling of the Lorenz attractor's phase space. This parallel processing structure facilitates the extraction of global dynamical features, enhancing the model's capacity to generalize across different regions of the attractor.
2. Temporal Correlation Preservation: The incorporation of d consecutive temporal measurements within each segment maintains the critical phase relationships between system variables (x, y, z). This temporal coherence is essential for capturing the underlying nonlinear dynamics that characterize the Lorenz system.

Mathematical Construction of the Network Architecture

Having established the dimensional characteristics of each component, we now formulate the complete transformation of the one-hidden-layer multilayer perceptron (MLP). The network architecture is characterized by two sequential transformations:

1. Input-to-Hidden Layer Transformation:

   $$\mathbf{H} = \mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)}$$

   where $\mathbf{X} \in \mathbb{R}^{n \times d}$ undergoes linear transformation through weight matrix $\mathbf{W}^{(1)} \in \mathbb{R}^{d \times h}$, followed by the addition of bias term $\mathbf{b}^{(1)} \in \mathbb{R}^{1 \times h}$, yielding hidden layer representation $\mathbf{H} \in \mathbb{R}^{n \times h}$.

2. Hidden-to-Output Layer Transformation:

$$\mathbf{O} = \mathbf{H}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}$$

where the hidden layer output $\mathbf{H}$ is mapped to the final output space through weight matrix $\mathbf{W}^{(2)} \in \mathbb{R}^{h \times q}$ and bias term $\mathbf{b}^{(2)} \in \mathbb{R}^{1 \times q}$, resulting in output $\mathbf{O} \in \mathbb{R}^{n \times q}$.

These transformations constitute the fundamental mathematical framework through which the network processes the input data, preserving the batch dimension n while implementing successive linear mappings between feature spaces.

While the introduction of a hidden layer appears to enhance network complexity through additional parametric dependencies, a careful mathematical analysis reveals an interesting property: the current architecture provides no additional representational capacity beyond that of a simple linear model. This phenomenon can be understood through examination of the network's compositional structure.

Consider the sequential transformations in our network:

1. The hidden layer implements an affine transformation of the input
2. The output layer similarly applies an affine transformation to the hidden layer representation

Through the properties of function composition, we can demonstrate that cascading affine transformations merely yields another affine transformation. Specifically, we can collapse the two-layer architecture into an equivalent single-layer representation:

$$\begin{aligned}\mathbf{O} &= (\mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)})\mathbf{W}^{(2)} + \mathbf{b}^{(2)} \\ &= \mathbf{X}\mathbf{W}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(2)} \\ &= \mathbf{X}\mathbf{W} + \mathbf{b}\end{aligned}$$

where:

$$\mathbf{W} = \mathbf{W}^{(1)}\mathbf{W}^{(2)} \quad \text{and} \quad \mathbf{b} = \mathbf{b}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}$$

This reduction demonstrates that the current architecture, despite its increased parametric complexity, remains functionally equivalent to a single-layer linear model. The additional parameters, while increasing the optimization space, do not expand the fundamental representational capacity of the network.

Nonlinear Activation and Enhanced Representational Capacity

To transcend the limitations of linear compositions, we introduce a crucial nonlinear element into the network architecture: the activation function σ. A prominent choice is the rectified linear unit (ReLU) function, defined element-wise as:

$$\sigma(x) = \max(0, x)$$

This nonlinear transformation fundamentally alters the network's computational structure. The modified architecture now takes the form:

$$\mathbf{H} = \sigma(\mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)})$$
$$\mathbf{O} = \mathbf{H}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}$$

The activation function σ operates on its inputs in a row-wise manner, processing each example in the batch independently. More precisely, the computation proceeds element-wise, allowing each hidden unit to operate autonomously without requiring information from other units within the same layer.

This architecture can be naturally extended to deeper configurations through iterative composition of hidden layers:

$$\mathbf{H}^{(1)} = \sigma_1(\mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)})$$
$$\mathbf{H}^{(2)} = \sigma_2(\mathbf{H}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(2)})$$

where distinct activation functions σ_1, σ_2 may be employed at different layers. This hierarchical structure enables the network to construct increasingly abstract representations, fundamentally expanding its capacity beyond simple affine transformations. Crucially, the introduction of nonlinearity prevents the type of layer collapse demonstrated in the purely linear case, ensuring that each additional layer contributes meaningful computational capacity to the network.

For detailed numerical implementation and specific computational protocols, we refer to the accompanying algorithmic frameworks in Julia programming language.