Implementation Logic and Mathematical Formulation of a Neural Network Visualization

Implementation Logic and Mathematical Formulation

1. Data Loading and Preprocessing

Data Loading and Preprocessing
- *Data Loading*: Data is read from the file `datasets.txt`, assuming each row contains two input features and one label.
- *Data Conversion*: The loaded data is converted into PyTorch tensors.
- Input feature matrix: $\mathbf{X} \in \mathbb{R}^{N \times 2}$, where N is the number of samples.
- Label vector: $\mathbf{y} \in \mathbb{R}^{N \times 1}$.

2. Model Definition

Model Definition
- *Neural Network Structure*: A two-layer fully connected neural network.
- *Hidden Layer*: Input dimension is 2, with H hidden neurons (adjustable parameter, e.g., `HIDDEN_SIZE`).
- *Output Layer*: Maps hidden layer output to 1 dimension.
- *Activation Function*: Sigmoid function used for both layers, defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

- *Weight Initialization*:
- Hidden layer weights $\mathbf{W}_h$ use Xavier normal initialization.
- Output layer weights $\mathbf{W}_o$ use standard normal initialization.
- *Mathematical Representation*:
- Hidden layer output:

$$\mathbf{h} = \sigma(\mathbf{W}_h \mathbf{x} + \mathbf{b}_h)$$

where $\mathbf{W}_h \in \mathbb{R}^{H \times 2}$, $\mathbf{b}_h \in \mathbb{R}^H$.
- Output layer output:

$$\hat{y} = \sigma(\mathbf{W}_o \mathbf{h} + \mathbf{b}_o)$$

where $\mathbf{W}_o \in \mathbb{R}^{1 \times H}$, $\mathbf{b}_o \in \mathbb{R}$.

3. Loss Function and Optimizer

Loss Function and Optimizer
- *Loss Function*: Binary Cross-Entropy Loss (BCELoss), defined as:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

- *Optimizer*: Stochastic Gradient Descent (SGD) with learning rate 3. Parameter update rule:

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$$

where $\theta = \{\mathbf{W}_h, \mathbf{b}_h, \mathbf{W}_o, \mathbf{b}_o\}$, $\eta = 3$.

4. Training Process

Training Process
- *Training Epochs*: 200 epochs.
- *Per Epoch Operations*:
1. *Forward Pass*: Compute hidden layer output $\mathbf{h}$ and model output $\hat{y}$.
2. *Loss Computation*: Calculate $\mathcal{L}$ based on $\hat{y}$ and $\mathbf{y}$.
3. *Backward Pass*: Compute gradients of $\mathcal{L}$ with respect to model parameters.
4. *Parameter Update*: Update θ using the optimizer.

5. Grid Generation

Grid Generation
- *Coarse Grid*: Generate a $G \times G$ grid in the input space [-1, 1] × [-1, 1] ($G =$ `GRID_SIZE`).
- Grid point coordinates:

$$\mathbf{G} = \{(x_i, y_j) \mid x_i = -1 + \frac{2i}{G-1}, y_j = -1 + \frac{2j}{G-1}, i, j = 0, \ldots, G-1\}$$

- *Fine Grid*: Generate a $F \times F$ grid in the same input space ($F =$ `FINE_GRID_SIZE`).
- Grid point coordinates:

$$\mathbf{F} = \{(x_k, y_l) \mid x_k = -1 + \frac{2k}{F-1}, y_l = -1 + \frac{2l}{F-1}, k, l = 0, \ldots, F-1\}$$

6. Hidden Layer Mapping

Hidden Layer Mapping
- *Coarse Grid Mapping*: Map coarse grid points $\mathbf{g} \in \mathbf{G}$ to the hidden space:

$$\mathbf{h}_g = \sigma(\mathbf{W}_h \mathbf{g} + \mathbf{b}_h)$$

- *Fine Grid Mapping*: Map fine grid points $\mathbf{f} \in \mathbf{F}$ to the hidden space:

$$\mathbf{h}_f = \sigma(\mathbf{W}_h \mathbf{f} + \mathbf{b}_h)$$

- *Model Output*: Compute model predictions for fine grid points:

$$\hat{y}_f = \sigma(\mathbf{W}_o \mathbf{h}_f + \mathbf{b}_o)$$

7. Visualization

Visualization
- *Static Plot*:
- Use fine grid hidden mappings $\mathbf{h}_f$ (first two dimensions) and model output $\hat{y}_f$ to plot the decision boundary ($\hat{y}_f = 0.5$).
- Plot training data points in the hidden space (based on $\mathbf{h}$).
- Use cubic spline interpolation to smoothly connect coarse grid mappings $\mathbf{h}_g$ as grid lines.
- *Animation*:
- Dynamically display changes in the hidden space across epochs, including:
- Movement of the decision boundary.

- Updates to scatter points $\mathbf{h}$.
- Smooth transitions of grid lines $\mathbf{h}_g$.

8. Mathematical Formula Summary

Mathematical Formula Summary
- *Model*:

$$\mathbf{h} = \sigma(\mathbf{W}_h \mathbf{x} + \mathbf{b}_h)$$
$$\hat{y} = \sigma(\mathbf{W}_o \mathbf{h} + \mathbf{b}_o)$$

- *Loss Function*:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

- *Optimization*:

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}, \quad \eta = 3$$

- *Grid Mapping*:

$$\mathbf{h}_g = \sigma(\mathbf{W}_h \mathbf{g} + \mathbf{b}_h), \quad \mathbf{g} \in \mathbf{G}$$

$$\mathbf{h}_f = \sigma(\mathbf{W}_h \mathbf{f} + \mathbf{b}_h), \quad \mathbf{f} \in \mathbf{F}$$

- *Decision Boundary*: In the hidden space, determined by $\hat{y}_f = 0.5$, i.e.:

$$\mathbf{W}_o \mathbf{h}_f + \mathbf{b}_o = 0$$

9. Spline Interpolation

Spline Interpolation
- *Cubic Spline Interpolation*: Used to smoothly connect coarse grid points in the hidden space $\mathbf{h}_g$.
- For each horizontal or vertical grid line, the interpolation function $s(t)$ satisfies $s(t_i) = \mathbf{h}_i$, with continuous second derivatives.

10. Animation Update

Animation Update
- *Per Frame Update*:
- Update decision boundary: Based on current $\mathbf{h}_f$ and $\hat{y}_f$.
- Update scatter points: Based on current $\mathbf{h}$.
- Update grid lines: Based on current $\mathbf{h}_g$, generating smooth curves via spline interpolation.