Computer Science & Information Systems

Deep Neural Networks - Lab sheet 5

Deep Feed-Forward Neural Network for Regression

Prepared by Seetha Parameswaran

1 Objective

The objective is to

- Understand deep feed-forward neural networks (DFNN) with multiple hidden layers.
- Implement forward propagation, backpropagation, and mini-batch SGD from scratch.
- Apply ReLU activation for hidden layers and identity activation for output.
- Use Mean Squared Error (MSE) loss for regression tasks.
- Predict hourly bike rental demand using Seoul Bike Sharing dataset.
- Implement He initialization and early stopping techniques.

2 Steps to be Performed

- **Tool:** Python3
- **Libraries required:** numpy, matplotlib, pandas, sklearn
- **Input:** Seoul Bike Sharing Demand Dataset (UCI repository)
- **Deep Learning Model:** Deep Feed-Forward Neural Network (DFNN)
- **ANN Architecture:** Multi-layer network [17, 64, 32, 16, 1] with ReLU hidden layers
- **Implementation:** L5_DFNN_for_Regression.ipynb

2.1 Steps

- Import required Python libraries (numpy, pandas, matplotlib, sklearn).
- Load the Seoul Bike Sharing Demand dataset (8,760 samples, 17 features).
- Understand the problem: Regression task to predict continuous bike rental count.
- Explore data: Visualize distribution, check for missing values, analyze features.
- Prepare the data: Extract features (weather, temporal) and target (bike count).

- Encode categorical variables: One-hot encode seasons, binary encode holiday/functioning day.
- Partition dataset into training (90%), validation (5%), and testing (5%) sets.
- Standardize features using StandardScaler (zero mean, unit variance).
- Standardize target variable for better optimization convergence.
- Define network architecture: 4 layers [17, 64, 32, 16, 1].
- Initialize parameters using He initialization for ReLU: $W^{(\ell)} \sim \mathcal{N}(0, \sqrt{2/n_{\ell-1}})$.
- Implement forward propagation with ReLU for hidden layers and identity for output.
- Compute Mean Squared Error (MSE) loss.
- Implement backpropagation using chain rule for gradient computation.
- Train model using mini-batch SGD with batch size 64.
- Apply early stopping: monitor validation loss, stop if no improvement for 15 epochs.
- Predict bike rental counts on test set.
- Inverse transform predictions to original scale.
- Compute evaluation metrics: MSE, RMSE, MAE, R^2 score.
- Visualize training history (loss curves for train and validation).
- Generate predictions vs actual plots and residual analysis.
- Analyze feature importance using first layer weights.

2.2 Mathematical Formulation

Network Architecture

$$\text{Input Layer:} \quad \mathbf{x} \in \mathbb{R}^{17} \tag{1}$$

$$\text{Hidden Layer 1:} \quad \mathbf{a}^{(1)} = \text{ReLU}(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}), \quad \mathbf{a}^{(1)} \in \mathbb{R}^{64} \tag{2}$$

$$\text{Hidden Layer 2:} \quad \mathbf{a}^{(2)} = \text{ReLU}(\mathbf{W}^{(2)}\mathbf{a}^{(1)} + \mathbf{b}^{(2)}), \quad \mathbf{a}^{(2)} \in \mathbb{R}^{32} \tag{3}$$

$$\text{Hidden Layer 3:} \quad \mathbf{a}^{(3)} = \text{ReLU}(\mathbf{W}^{(3)}\mathbf{a}^{(2)} + \mathbf{b}^{(3)}), \quad \mathbf{a}^{(3)} \in \mathbb{R}^{16} \tag{4}$$

$$\text{Output Layer:} \quad \hat{y} = \mathbf{W}^{(4)}\mathbf{a}^{(3)} + \mathbf{b}^{(4)}, \quad \hat{y} \in \mathbb{R} \tag{5}$$

Activation Functions

$$\text{ReLU (Hidden Layers):} \quad f(z) = \max(0, z) \tag{6}$$

$$\text{ReLU Derivative:} \quad f'(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

$$\text{Identity (Output):} \quad f(z) = z, \quad f'(z) = 1 \tag{8}$$

Loss Function

$$\text{Mean Squared Error:} \quad J(\mathbf{\Theta}) = \frac{1}{2N}\sum_{i=1}^{N}(\hat{y}^{(i)} - y^{(i)})^2 \tag{9}$$

where $\mathbf{\Theta} = \{\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \mathbf{W}^{(2)}, \mathbf{b}^{(2)}, \mathbf{W}^{(3)}, \mathbf{b}^{(3)}, \mathbf{W}^{(4)}, \mathbf{b}^{(4)}\}$.

Forward Propagation (Vectorized for Mini-batch)

$$\mathbf{Z}^{(1)} = \mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)} \quad \text{(Linear transformation)} \tag{10}$$
$$\mathbf{A}^{(1)} = \text{ReLU}(\mathbf{Z}^{(1)}) \quad \text{(Non-linear activation)} \tag{11}$$
$$\mathbf{Z}^{(2)} = \mathbf{A}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(2)} \tag{12}$$
$$\mathbf{A}^{(2)} = \text{ReLU}(\mathbf{Z}^{(2)}) \tag{13}$$
$$\mathbf{Z}^{(3)} = \mathbf{A}^{(2)}\mathbf{W}^{(3)} + \mathbf{b}^{(3)} \tag{14}$$
$$\mathbf{A}^{(3)} = \text{ReLU}(\mathbf{Z}^{(3)}) \tag{15}$$
$$\mathbf{Z}^{(4)} = \mathbf{A}^{(3)}\mathbf{W}^{(4)} + \mathbf{b}^{(4)} \tag{16}$$
$$\hat{\mathbf{Y}} = \mathbf{Z}^{(4)} \quad \text{(Identity activation)} \tag{17}$$

where $\mathbf{X} \in \mathbb{R}^{B\times 17}$ is a mini-batch of B samples.

Backpropagation (Gradient Computation)

Output Layer Gradient:

$$\frac{\partial J}{\partial \mathbf{Z}^{(4)}} = \frac{1}{B}(\hat{\mathbf{Y}} - \mathbf{Y}) \tag{18}$$
$$\frac{\partial J}{\partial \mathbf{W}^{(4)}} = (\mathbf{A}^{(3)})^T \frac{\partial J}{\partial \mathbf{Z}^{(4)}} \tag{19}$$
$$\frac{\partial J}{\partial \mathbf{b}^{(4)}} = \sum_{i=1}^{B} \frac{\partial J}{\partial z_i^{(4)}} \tag{20}$$

Hidden Layer 3 Gradient:

$$\frac{\partial J}{\partial \mathbf{A}^{(3)}} = \frac{\partial J}{\partial \mathbf{Z}^{(4)}}(\mathbf{W}^{(4)})^T \tag{21}$$
$$\frac{\partial J}{\partial \mathbf{Z}^{(3)}} = \frac{\partial J}{\partial \mathbf{A}^{(3)}} \odot f'(\mathbf{Z}^{(3)}) \quad \text{(Element-wise ReLU derivative)} \tag{22}$$
$$\frac{\partial J}{\partial \mathbf{W}^{(3)}} = (\mathbf{A}^{(2)})^T \frac{\partial J}{\partial \mathbf{Z}^{(3)}} \tag{23}$$
$$\frac{\partial J}{\partial \mathbf{b}^{(3)}} = \sum_{i=1}^{B} \frac{\partial J}{\partial z_i^{(3)}} \tag{24}$$

Hidden Layer 2 Gradient:

$$\frac{\partial J}{\partial \mathbf{A}^{(2)}} = \frac{\partial J}{\partial \mathbf{Z}^{(3)}}(\mathbf{W}^{(3)})^T \tag{25}$$

$$\frac{\partial J}{\partial \mathbf{Z}^{(2)}} = \frac{\partial J}{\partial \mathbf{A}^{(2)}} \odot f'(\mathbf{Z}^{(2)}) \tag{26}$$

$$\frac{\partial J}{\partial \mathbf{W}^{(2)}} = (\mathbf{A}^{(1)})^T \frac{\partial J}{\partial \mathbf{Z}^{(2)}} \tag{27}$$

$$\frac{\partial J}{\partial \mathbf{b}^{(2)}} = \sum_{i=1}^{B} \frac{\partial J}{\partial z_i^{(2)}} \tag{28}$$

Hidden Layer 1 Gradient:

$$\frac{\partial J}{\partial \mathbf{A}^{(1)}} = \frac{\partial J}{\partial \mathbf{Z}^{(2)}}(\mathbf{W}^{(2)})^T \tag{29}$$

$$\frac{\partial J}{\partial \mathbf{Z}^{(1)}} = \frac{\partial J}{\partial \mathbf{A}^{(1)}} \odot f'(\mathbf{Z}^{(1)}) \tag{30}$$

$$\frac{\partial J}{\partial \mathbf{W}^{(1)}} = \mathbf{X}^T \frac{\partial J}{\partial \mathbf{Z}^{(1)}} \tag{31}$$

$$\frac{\partial J}{\partial \mathbf{b}^{(1)}} = \sum_{i=1}^{B} \frac{\partial J}{\partial z_i^{(1)}} \tag{32}$$

Parameter Update (Mini-batch SGD)

$$\mathbf{W}^{(\ell)} := \mathbf{W}^{(\ell)} - \eta \frac{\partial J}{\partial \mathbf{W}^{(\ell)}}, \quad \ell = 1, 2, 3, 4 \tag{33}$$

$$\mathbf{b}^{(\ell)} := \mathbf{b}^{(\ell)} - \eta \frac{\partial J}{\partial \mathbf{b}^{(\ell)}}, \quad \ell = 1, 2, 3, 4 \tag{34}$$

where η is the learning rate (typically 0.001).

Evaluation Metrics

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \hat{y}^{(i)})^2 \tag{35}$$

$$\text{RMSE} = \sqrt{\text{MSE}} \tag{36}$$

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y^{(i)} - \hat{y}^{(i)}| \tag{37}$$

$$\text{R}^2 = 1 - \frac{\sum_{i=1}^{N}(y^{(i)} - \hat{y}^{(i)})^2}{\sum_{i=1}^{N}(y^{(i)} - \bar{y})^2} \tag{38}$$

3 Results

- Deep feed-forward neural network successfully trained with 4 layers.

- Total parameters: 3,777 (significantly more capacity than linear regression).
- Model achieved Test RMSE ≈ 335 bikes/hour on Seoul bike dataset.
- Test MAE ≈ 238 bikes/hour indicates average prediction error.
- Test R^2 score ≈ 0.72 shows model explains 72% of variance.
- Training converged in ≈ 50-60 epochs with early stopping.
- Validation loss decreased smoothly, indicating no overfitting.
- ReLU activation prevented vanishing gradient problem in deep network.
- He initialization enabled stable training from the start.
- Mini-batch SGD (batch size 64) balanced speed and stability.
- Model predictions closely follow actual demand patterns.
- Feature importance analysis shows Solar Radiation and Wind Speed as top predictors.

4 Observation

- Multiple hidden layers enable learning of complex non-linear relationships.
- DFNN significantly outperforms linear regression on this dataset.
- ReLU activation is computationally efficient and prevents vanishing gradients.
- Identity activation in output layer allows continuous value predictions.
- Backpropagation efficiently computes gradients through multiple layers using chain rule.
- Mini-batch SGD provides good balance: faster than batch GD, more stable than SGD.
- Feature standardization is critical for fast and stable convergence in deep networks.
- Early stopping prevents overfitting by monitoring validation loss.
- He initialization (designed for ReLU) maintains gradient scale through layers.
- Progressive dimensionality reduction (17→64→32→16→1) works well for this problem.
- Model captures temporal patterns (hour of day) and weather dependencies effectively.
- Residual plot shows relatively random scatter, indicating good model fit.
- Deeper networks require more careful hyperparameter tuning than shallow models.
- Network provides hierarchical feature learning: low→mid→high level patterns.
- Model suitable for smart city applications requiring real-time demand prediction.

5 Key Differences from Linear Regression

- **Architecture:** Multiple hidden layers vs single layer.
- **Non-linearity:** ReLU activations vs no activation (identity only).
- **Capacity:** 3,777 parameters vs ≈ 18 parameters.
- **Learning:** Backpropagation through layers vs direct gradient.
- **Expressiveness:** Can model complex non-linear functions vs linear only.
- **Training time:** Longer training but better performance.
- **Initialization:** He initialization critical for deep networks.
- **Optimization:** Mini-batch SGD with early stopping vs simple GD.

6 Applications

- **Smart Cities:** Real-time bike-sharing demand prediction and rebalancing.
- **Transportation:** Traffic flow prediction, ride-sharing demand forecasting.
- **Energy:** Electricity load forecasting, renewable energy prediction.
- **Finance:** Stock price prediction, portfolio optimization.
- **Healthcare:** Patient readmission prediction, treatment response modeling.
- **Retail:** Sales forecasting, inventory optimization.
- **Manufacturing:** Predictive maintenance, quality control.
- **Weather:** Temperature prediction, precipitation forecasting.