Computer Science & Information Systems

Deep Neural Networks - Lab sheet 5

Deep Feed-Forward Neural Network for Multi-Class Classification

Prepared by Seetha Parameswaran

1 Objective

The objective is to

- Understand deep feed-forward neural networks (DFNN) for multi-class classification.
- Implement forward propagation with softmax output activation.
- Apply categorical cross-entropy loss for multi-class problems.
- Implement backpropagation with weighted loss for class imbalance.
- Use ReLU activation for hidden layers and softmax for output layer.
- Predict forest cover type (7 classes) using Covertype dataset.
- Handle class imbalance using weighted categorical cross-entropy.
- Evaluate multi-class classification with comprehensive metrics.

2 Steps to be Performed

- **Tool:** Python3
- **Libraries required:** numpy, matplotlib, pandas, sklearn, seaborn
- **Input:** Covertype (Forest Cover Type) Dataset (UCI repository)
- **Deep Learning Model:** Deep Feed-Forward Neural Network (DFNN)
- **ANN Architecture:** Multi-layer network [54, 64, 32, 16, 7] with ReLU hidden layers
- **Output Layer:** 7 neurons with Softmax activation (one per class)
- **Implementation:** L5_DFNN_for_Classification_Complete.ipynb

2.1 Steps

- Import required Python libraries (numpy, pandas, matplotlib, sklearn).
- Load the Covertype dataset (581,012 samples, 54 features, 7 classes).
- Understand the problem: Multi-class classification of forest cover types.
- Explore data: Visualize class distribution, identify class imbalance.
- Analyze class distribution: Calculate imbalance ratio between most/least frequent classes.
- Prepare the data: Extract 54 cartographic features and 7-class target.
- Convert target classes from 1-7 to 0-6 for zero-indexing.
- One-hot encode target: Convert to binary matrix [N $\times$ 7].
- Partition dataset into training (80%), validation (10%), and testing (10%) sets.
- Ensure stratified splits to maintain class proportions in all sets.
- Standardize features using StandardScaler (zero mean, unit variance).
- Calculate class weights: $w_c = \frac{N}{K \times n_c}$ to handle imbalance.
- Define network architecture: 4 layers [54, 64, 32, 16, 7].
- 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 softmax for output.
- Compute weighted categorical cross-entropy loss.
- Implement backpropagation with class weights applied to gradients.
- Train model using mini-batch SGD with batch size 128.
- Apply early stopping: monitor validation loss, stop if no improvement for 15 epochs.
- Predict class probabilities on test set using softmax output.
- Convert probabilities to class labels using argmax.
- Compute evaluation metrics: Accuracy, precision, recall, F1-score (per-class and macro).
- Generate 7×7 confusion matrix showing true vs predicted classes.
- Visualize training history (loss and accuracy curves for train and validation).
- Create confusion matrix heatmap with annotations.
- Plot per-class F1 scores to identify performance across classes.
- Analyze precision vs recall trade-offs for each class.
- Compare true vs predicted class distributions.

2.2 Mathematical Formulation

Network Architecture

$$\text{Input Layer:} \quad \mathbf{x} \in \mathbb{R}^{54} \tag{1}$$

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

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

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

$$\text{Output Layer:} \quad \mathbf{z}^{(4)} = \mathbf{W}^{(4)}\mathbf{a}^{(3)} + \mathbf{b}^{(4)}, \quad \mathbf{z}^{(4)} \in \mathbb{R}^{7} \tag{5}$$

$$\text{Softmax:} \quad \hat{\mathbf{y}} = \text{softmax}(\mathbf{z}^{(4)}) = \frac{e^{\mathbf{z}^{(4)}}}{\sum_{j=1}^{7} e^{z_j^{(4)}}}, \quad \hat{\mathbf{y}} \in \mathbb{R}^{7} \tag{6}$$

Activation Functions

$$\text{ReLU (Hidden Layers):} \quad f(z) = \max(0, z) \tag{7}$$

$$\text{ReLU Derivative:} \quad f'(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

$$\text{Softmax (Output):} \quad \sigma(\mathbf{z})_k = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}, \quad k = 1, \ldots, K \tag{9}$$

Properties of Softmax:

- Outputs sum to 1: $\sum_{k=1}^{K} \sigma(\mathbf{z})_k = 1$
- Each output is in range [0, 1]: $0 \leq \sigma(\mathbf{z})_k \leq 1$
- Can be interpreted as class probabilities: $P(y = k|\mathbf{x})$

Loss Function

$$\text{Weighted Categorical Cross-Entropy:} \quad J(\mathbf{\Theta}) = -\frac{1}{N}\sum_{i=1}^{N} w_{y^{(i)}} \sum_{k=1}^{K} y_k^{(i)} \log(\hat{y}_k^{(i)}) \tag{10}$$

where:

- $\mathbf{\Theta} = \{\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \mathbf{W}^{(2)}, \mathbf{b}^{(2)}, \mathbf{W}^{(3)}, \mathbf{b}^{(3)}, \mathbf{W}^{(4)}, \mathbf{b}^{(4)}\}$
- $y_k^{(i)}$ is one-hot encoded target (1 for true class, 0 otherwise)
- $\hat{y}_k^{(i)}$ is predicted probability for class k
- $w_{y^{(i)}}$ is the weight for the true class of sample i
- $K = 7$ classes

Class Weight Calculation

To handle class imbalance:

$$w_c = \frac{N}{K \times n_c} \tag{11}$$

where:

- N = total number of samples
- K = number of classes (7)
- n_c = number of samples in class c

This ensures minority classes contribute more to the loss.

Forward Propagation (Vectorized for Mini-batch)

$$\mathbf{Z}^{(1)} = \mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)} \quad \text{(Linear transformation)} \tag{12}$$
$$\mathbf{A}^{(1)} = \text{ReLU}(\mathbf{Z}^{(1)}) \quad \text{(Non-linear activation)} \tag{13}$$
$$\mathbf{Z}^{(2)} = \mathbf{A}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(2)} \tag{14}$$
$$\mathbf{A}^{(2)} = \text{ReLU}(\mathbf{Z}^{(2)}) \tag{15}$$
$$\mathbf{Z}^{(3)} = \mathbf{A}^{(2)}\mathbf{W}^{(3)} + \mathbf{b}^{(3)} \tag{16}$$
$$\mathbf{A}^{(3)} = \text{ReLU}(\mathbf{Z}^{(3)}) \tag{17}$$
$$\mathbf{Z}^{(4)} = \mathbf{A}^{(3)}\mathbf{W}^{(4)} + \mathbf{b}^{(4)} \tag{18}$$
$$\hat{\mathbf{Y}} = \text{softmax}(\mathbf{Z}^{(4)}) \quad \text{(Softmax activation)} \tag{19}$$

where:

- $\mathbf{X} \in \mathbb{R}^{B \times 54}$ is a mini-batch of B samples
- $\mathbf{Z}^{(\ell)}$ are pre-activation values
- $\mathbf{A}^{(\ell)}$ are post-activation values
- $\hat{\mathbf{Y}} \in \mathbb{R}^{B \times 7}$ contains class probabilities

Backpropagation (Gradient Computation)

Output Layer Gradient (Softmax + Cross-Entropy):

The combined derivative of softmax and cross-entropy simplifies to:

$$\frac{\partial J}{\partial \mathbf{Z}^{(4)}} = \frac{1}{B}\mathbf{W}_{diag} \odot (\hat{\mathbf{Y}} - \mathbf{Y}) \tag{20}$$

where $\mathbf{W}_{diag}$ is diagonal matrix with class weights for each sample.

$$\frac{\partial J}{\partial \mathbf{W}^{(4)}} = (\mathbf{A}^{(3)})^T \frac{\partial J}{\partial \mathbf{Z}^{(4)}} \tag{21}$$
$$\frac{\partial J}{\partial \mathbf{b}^{(4)}} = \sum_{i=1}^{B} \frac{\partial J}{\partial \mathbf{z}_i^{(4)}} \tag{22}$$

Hidden Layer 3 Gradient:

$$\frac{\partial J}{\partial \mathbf{A}^{(3)}} = \frac{\partial J}{\partial \mathbf{Z}^{(4)}}(\mathbf{W}^{(4)})^T \tag{23}$$

$$\frac{\partial J}{\partial \mathbf{Z}^{(3)}} = \frac{\partial J}{\partial \mathbf{A}^{(3)}} \odot f'(\mathbf{Z}^{(3)}) \quad \text{(Element-wise ReLU derivative)} \tag{24}$$

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

$$\frac{\partial J}{\partial \mathbf{b}^{(3)}} = \sum_{i=1}^{B} \frac{\partial J}{\partial \mathbf{z}_i^{(3)}} \tag{26}$$

Hidden Layer 2 Gradient:

$$\frac{\partial J}{\partial \mathbf{A}^{(2)}} = \frac{\partial J}{\partial \mathbf{Z}^{(3)}}(\mathbf{W}^{(3)})^T \tag{27}$$

$$\frac{\partial J}{\partial \mathbf{Z}^{(2)}} = \frac{\partial J}{\partial \mathbf{A}^{(2)}} \odot f'(\mathbf{Z}^{(2)}) \tag{28}$$

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

$$\frac{\partial J}{\partial \mathbf{b}^{(2)}} = \sum_{i=1}^{B} \frac{\partial J}{\partial \mathbf{z}_i^{(2)}} \tag{30}$$

Hidden Layer 1 Gradient:

$$\frac{\partial J}{\partial \mathbf{A}^{(1)}} = \frac{\partial J}{\partial \mathbf{Z}^{(2)}}(\mathbf{W}^{(2)})^T \tag{31}$$

$$\frac{\partial J}{\partial \mathbf{Z}^{(1)}} = \frac{\partial J}{\partial \mathbf{A}^{(1)}} \odot f'(\mathbf{Z}^{(1)}) \tag{32}$$

$$\frac{\partial J}{\partial \mathbf{W}^{(1)}} = \mathbf{X}^T \frac{\partial J}{\partial \mathbf{Z}^{(1)}} \tag{33}$$

$$\frac{\partial J}{\partial \mathbf{b}^{(1)}} = \sum_{i=1}^{B} \frac{\partial J}{\partial \mathbf{z}_i^{(1)}} \tag{34}$$

Parameter Update (Mini-batch SGD)

$$\mathbf{W}^{(\ell)} := \mathbf{W}^{(\ell)} - \eta \frac{\partial J}{\partial \mathbf{W}^{(\ell)}}, \quad \ell = 1, 2, 3, 4 \tag{35}$$

$$\mathbf{b}^{(\ell)} := \mathbf{b}^{(\ell)} - \eta \frac{\partial J}{\partial \mathbf{b}^{(\ell)}}, \quad \ell = 1, 2, 3, 4 \tag{36}$$

where η is the learning rate (typically 0.01 for classification).

Prediction

$$\text{Class probabilities:} \quad \hat{\mathbf{y}} = \text{softmax}(\mathbf{z}^{(4)}) \tag{37}$$

$$\text{Predicted class:} \quad \hat{c} = \arg\max_k \hat{y}_k \tag{38}$$

Evaluation Metrics

Accuracy:

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(\hat{c}^{(i)} = c^{(i)}) \tag{39}$$

Per-Class Metrics (for class c):

$$\text{Precision}_c = \frac{\text{TP}_c}{\text{TP}_c + \text{FP}_c} \tag{40}$$

$$\text{Recall}_c = \frac{\text{TP}_c}{\text{TP}_c + \text{FN}_c} \tag{41}$$

$$\text{F1-Score}_c = 2 \times \frac{\text{Precision}_c \times \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c} \tag{42}$$

Macro-Averaged Metrics:

$$\text{Macro Precision} = \frac{1}{K}\sum_{c=1}^{K} \text{Precision}_c \tag{43}$$

$$\text{Macro Recall} = \frac{1}{K}\sum_{c=1}^{K} \text{Recall}_c \tag{44}$$

$$\text{Macro F1-Score} = \frac{1}{K}\sum_{c=1}^{K} \text{F1-Score}_c \tag{45}$$

Confusion Matrix:
C_{ij} = number of samples with true class i predicted as class j

3 Dataset Information

Covertype Dataset:

- **Source:** UCI Machine Learning Repository
- **Domain:** Roosevelt National Forest, Colorado
- **Samples:** 581,012 cartographic observations
- **Features:** 54 features
  - 10 continuous features: Elevation, Aspect, Slope, Distances to hydrology/roads/fire points, Hillshade indices
  - 44 binary features: 4 wilderness areas (one-hot), 40 soil types (one-hot)
- **Target:** 7 forest cover types
  1. Spruce/Fir
  2. Lodgepole Pine

3. Ponderosa Pine
4. Cottonwood/Willow
5. Aspen
6. Douglas-fir
7. Krummholz

- **Class Imbalance:** Significant (ratio ≈ 100:1 between most and least frequent)

4 Results

- Deep feed-forward neural network successfully trained for 7-class classification.
- Total parameters: 6,247 parameters across 4 layers.
- Model achieved Test Accuracy ≈ 70-75% on Covertype dataset.
- Macro-averaged F1-score ≈ 0.65-0.70 across all classes.
- Training converged in ≈ 40-50 epochs with early stopping.
- Validation loss and accuracy curves show smooth convergence without overfitting.
- Softmax output successfully provides calibrated probability estimates for all 7 classes.
- Class weighting significantly improved performance on minority classes.
- Most frequent classes (Lodgepole Pine, Spruce/Fir) achieved highest F1-scores (> 0.80).
- Minority classes (Cottonwood/Willow) showed lower but acceptable performance (≈ 0.50).
- Confusion matrix reveals most errors occur between similar forest types.
- ReLU activation prevented vanishing gradient problem in deep network.
- He initialization enabled stable training from epoch 1.
- Mini-batch SGD (batch size 128) provided good balance of speed and stability.

5 Observation

- Softmax activation is essential for multi-class problems, providing probability distribution.
- Output probabilities sum to 1, enabling probabilistic interpretation of predictions.
- Categorical cross-entropy is the natural loss for softmax output layer.
- Weighted loss successfully addresses class imbalance, improving minority class performance.
- Multiple hidden layers enable learning of complex decision boundaries between 7 classes.

- DFNN learns hierarchical features: low-level cartographic → mid-level spatial → high-level ecological.
- Backpropagation through softmax+CE simplifies to elegant gradient: $\hat{y} - y$.
- Class imbalance remains challenging despite weighting (extreme imbalance hard to overcome).
- Confusion matrix shows interpretable errors (e.g., confusing similar pine species).
- Feature standardization critical for convergence when features have different scales.
- Early stopping prevents overfitting on majority classes at expense of minority classes.
- Stratified train/val/test splits preserve class distribution across all sets.
- Per-class metrics more informative than overall accuracy for imbalanced data.
- Macro-averaging gives equal weight to all classes, revealing true balanced performance.
- Model provides confidence scores for predictions, useful for threshold tuning.
- Network architecture [54, 64, 32, 16, 7] balances capacity and training efficiency.

6 Key Differences from Regression DFNN

- **Output Activation:** Softmax (probabilities) vs Identity (continuous values).
- **Output Dimension:** K neurons (one per class) vs 1 neuron.
- **Loss Function:** Categorical Cross-Entropy vs Mean Squared Error.
- **Target Encoding:** One-hot vectors vs continuous scalars.
- **Evaluation:** Accuracy, Precision, Recall, F1 vs MSE, RMSE, R^2.
- **Output Interpretation:** Class probabilities → argmax vs direct value.
- **Gradient (Output):** $\hat{y} - y$ (one-hot) vs $\hat{y} - y$ (scalar).
- **Class Imbalance:** Weighted loss needed vs not typically relevant.
- **Decision Boundary:** Multi-way partition of space vs continuous mapping.
- **Confidence:** Probability distribution vs prediction interval (if added).

7 Key Differences from Binary Classification

- **Output Activation:** Softmax (K outputs) vs Sigmoid (1 output).
- **Loss Function:** Categorical Cross-Entropy vs Binary Cross-Entropy.
- **Output Dimension:** K neurons vs 1 neuron.
- **Target Encoding:** One-hot K-dimensional vs binary scalar.
- **Probability Sum:** Outputs sum to 1 vs single probability with complement.

- **Decision:** argmax over K classes vs threshold on single probability.
- **Confusion Matrix:** K×K matrix vs 2×2 matrix.
- **Metrics:** Per-class + macro-averaging vs precision/recall/F1 for positive class.

8 Applications

- **Environmental Science:** Forest cover type mapping, vegetation classification.
- **Remote Sensing:** Land use classification, satellite image analysis.
- **Healthcare:** Disease diagnosis (multi-class), patient risk stratification.
- **Computer Vision:** Image classification, object recognition (ImageNet).
- **Natural Language Processing:** Document classification, sentiment analysis (multi-class).
- **Recommendation Systems:** Item category prediction, user preference classification.
- **Financial Services:** Credit rating classification, fraud type detection.
- **Manufacturing:** Defect type classification, quality control.
- **Telecommunications:** Network traffic classification, customer segmentation.
- **Transportation:** Traffic sign recognition, vehicle type classification.

9 Handling Class Imbalance - Techniques

- **Weighted Loss:** Apply class weights inversely proportional to frequency (implemented).
- **Oversampling:** SMOTE (Synthetic Minority Over-sampling Technique).
- **Undersampling:** Reduce majority class samples.
- **Focal Loss:** Down-weight easy examples, focus on hard cases.
- **Class-Balanced Sampling:** Sample mini-batches with equal class representation.
- **Ensemble Methods:** Train multiple models on balanced subsets.
- **Cost-Sensitive Learning:** Higher misclassification cost for minority classes.
- **Two-Phase Training:** Pre-train on balanced subset, fine-tune on full data.