ReLU Neural Networks

Locally Interpretable Deep Learning

July 7, 2025

Introduction to ReLU Neural Networks

- Deep Neural Networks (DNNs) with ReLU activation functions
- Powerful models for complex pattern learning
- Often considered “black box” models due to lack of transparency
- Can be made interpretable through local linear representations
- L1 regularization can reduce complexity
- MoDeVa provides tools for transparency while maintaining performance

ReLU DNN Model Architecture

Mathematical Formulation

Feedforward neural network with:

- Inputs $\mathbf{x} \in \mathbb{R}^d$, L hidden layers, one output neuron
- l-th hidden layer has n_l neurons
- Weight matrix: $\mathbf{W}^{(l)}$ of size $n_{l+1} \times n_l$
- Bias vector: $\mathbf{b}^{(l)}$ of size n_{l+1}

Network Equations

$$\mathbf{z}^{(l+1)} = \mathbf{W}^{(l)}\chi^{(l)} + \mathbf{b}^{(l)} \tag{1}$$

$$\chi^{(l)} = \max(0, \mathbf{z}^{(l)}) \quad \text{(ReLU activation)} \tag{2}$$

$$f(\mathbf{x}) = \sigma(\mathbf{W}^{(L)}\chi^{(L)} + \mathbf{b}^{(L)}) \quad \text{(Output layer)} \tag{3}$$

σ is identity (regression) or sigmoid (binary classification)

ReLU DNN Model Architecture

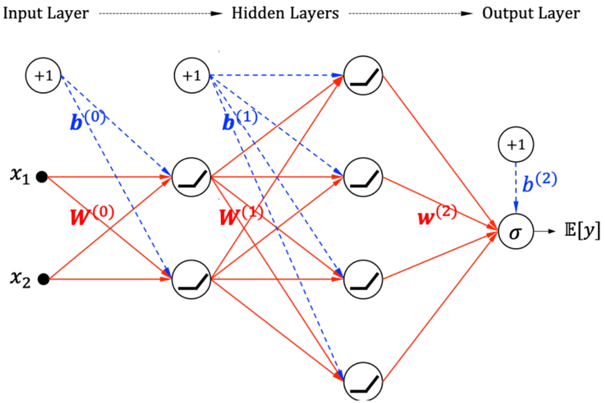

Figure: 2-Layer ReLU DNN

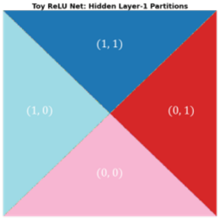

Figure: 1st Layer Partition

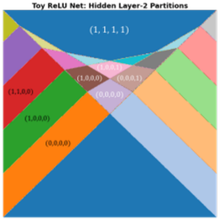

Figure: 2nd Layer Partition

Local Linear Models - Unwrapping the Black Box

Activation Pattern

Binary vector $C = [C^{(1)}; \ldots; C^{(L)}]$ indicates on/off state of each hidden neuron:

$$C(\mathbf{x}) = [I(\mathbf{z}_1^{(1)} > 0); \ldots; I(\mathbf{z}_{n_L}^{(L)} > 0)] \tag{4}$$

Key Insight

- Samples with same activation pattern can be grouped
- Their input-output relationship can be simplified using a linear model

Local Linear Model (LLM)

$$f(\mathbf{x}) = \tilde{\mathbf{w}}^{C(\mathbf{x})T}\mathbf{x} + \tilde{b}^{C(\mathbf{x})} \tag{5}$$

- $\tilde{\mathbf{w}}^{C(\mathbf{x})}$ - Coefficients of the linear model

Data Preparation for ReLU DNN Models

```python
from modeva import DataSet
ds = DataSet()
ds.load(name="BikeSharing")

# Preprocessing steps
ds.scale_numerical(features=("cnt",), method="log1p")
ds.scale_numerical(features=ds.feature_names,
                   method="minmax")
ds.set_inactive_features(features=("yr", "season", "temp"))
ds.preprocess()

# Split data into training and testing sets
ds.set_random_split()
```

Important Preprocessing Steps

- Feature scaling is crucial for neural networks
- MinMax scaling preserves the range of values
- Proper preprocessing enhances model training and convergence

Model Configuration in MoDeVa

```
# For regression tasks
from modeva.models import MoReLUDNNRegressor
model_relunet = MoReLUDNNRegressor(name="ReLU_Net",
                                   hidden_layer_sizes=(40, 40),
                                   l1_reg=0.0002,
                                   learning_rate=0.001)

# For classification tasks
from modeva.models import MoReLUDNNClassifier
model_relunet = MoReLUDNNClassifier(name="ReLU_Net",
                                   hidden_layer_sizes=(40, 40),
                                   l1_reg=0.0002,
                                   learning_rate=0.001)
```

Training and Evaluation

```
# Train model
model_relunet.fit(ds.train_x, ds.train_y.ravel())

# Create TestSuite for evaluation
from modeva import TestSuite
```

Important Hyperparameters for ReLU DNN

hidden_layer_sizes

- Tuple specifying hidden layer structure
- Small networks: Limited expressive power
- Large networks: Too complex to interpret
- Recommendation: Start larger, then apply L1 penalty

l1_reg

- Regularization strength (default: 1e-5)
- Shrinks weights toward zero
- Bias terms remain unpenalized
- Avoids overfitting
- Enhances interpretability
- Higher values reduce number of LLMs

learning_rate

- Controls step size of gradient descent (default: 0.001)
- Critical for model performance
- Small values: Longer training time
- Large values: Unstable training

Understanding Local Linear Models

```
# Summary of local linear models
result = ts.interpret_llm_summary()
result.table
```

Table Components

- **count**: Number of training samples in each LLM
- **Response Mean**: Average response values
- **Local AUC**: Performance in the LLM's region
- **Global AUC**: Performance on all samples

Interpretation Value

- Understanding trained ReLU-DNN
- Identifying effective LLMs
- Comparing local vs. global performance
- May indicate when simpler models suffice

Visualizing LLM Coefficients

```python
# Parallel coordinate of LLM coefficients
result = ts.interpret_llm_pc()
result.plot()
```

Visualization Features

- Each line represents a single LLM
- X-axis: Feature names
- Y-axis: Coefficient values
- Typically shows top 10 important features

Interpretation Guidelines

- Large coefficient values: Important features
- Positive coefficients: Monotonic increasing effect
- Coefficients near zero: Trivial features
- Wide coefficient range: Nonlinear effect

Global Feature Importance

```python
# Global feature importance
result = ts.interpret_fi()
result.plot()
```

Calculation Method

- Calculate squared sum of LLM coefficients per feature
- Normalize importance values to sum to one

Visualization

- Bar chart
- Features in descending order of importance
- Relative contribution to model predictions

Practical Applications

- Feature selection
- Dimension reduction
- Feature engineering guidance
- Model simplification

Understanding Feature Effects

```python
# LLM profile plot for specific feature
result = ts.interpret_llm_profile(features="hr")
result.plot()
```

Visualization Elements

- Each line: One LLM
- X-axis: Feature values
- Y-axis: Marginal effect
- Typically shows top 30 LLMs
- Effects are de-meaned

Interpretation Value

- How features affect predictions across LLMs
- Range and direction of effects
- Potential nonlinear relationships
- Feature interaction patterns
- Regions of high variability

Individual Prediction Analysis

```python
# Local feature importance for specific sample
result = ts.local_linear_fi(sample_index=10,
                            centered=True)
result.plot()
```

Visualization Components

- **Stem**: Direction and magnitude to prediction
- **Bar**: Direction and magnitude of effects
- **Feature values**: Values for the specific sample
- **Comparison**: Reference to average behavior

Centering Options

- **Uncentered** (centered=False):
  - Raw feature contributions
  - Direct interpretation
  - May have identifiability issues
- **Centered** (centered=True):
  - Compares to population mean
  - More stable interpretation
  - Better for relative importance

Benefits of Interpretable ReLU DNNs

Transparency Benefits

- Transforms "black box" into set of interpretable linear models
- Reveals which features matter and how they influence predictions
- Identifies how predictions are made in different regions

Technical Benefits

- L1 regularization reduces model complexity while maintaining performance
- Balances predictive power with interpretability
- Facilitates model selection and comparison

Addressing Interpretability-Performance Tradeoff

- Traditional view: Interpretability reduces performance
- ReLU DNN with LLM: Maintains high performance while adding transparency

When to Use ReLU DNNs with LLM Interpretation

Ideal Use Cases

- Complex, high-dimensional datasets with potential nonlinearities
- Applications requiring both high accuracy and interpretability
- Regulatory environments where model transparency is mandated
- Knowledge discovery tasks where understanding relationships is important
- When exact local explanations are needed

When to Consider Alternatives

- When a single LLM dominates (try simpler linear models)
- When a few LLMs are sufficient, Neural Tree might be better
- Low-dimensional problems

Tips for Effective ReLU DNN Implementation

Model Building

- Start with larger network, then apply L1 regularization
- Tune L1 regularization to balance complexity and performance
- Always scale features for optimal performance
- Carefully tune learning rate for stable convergence
- Refine based on insights gained from interpretability

Interpretation Strategy

- Use both global and local interpretation methods
- Analyze which LLMs perform best in which regions
- Look for patterns across multiple LLMs