Neural Networks for Binary Classification Using Softmax

Qingfeng Liu

Hosei University

August 3, 2025

Practical Application Example: Spam Email Classification

System Design

- **Input**: Email text feature vector (word frequencies, etc.)
- **Output**:
  - Class 0: Normal email (probability P0)
  - Class 1: Spam email (probability P1)

Practical Application Example: Spam Email Classification

Example Calculation

For email containing "free", "prize":

- Output scores: [2.3, 5.1] (normal vs spam)
- Calculations:

$$e^{2.3} \approx 9.9742$$
$$e^{5.1} \approx 164.022$$
$$P0 = \frac{9.9742}{9.9742 + 164.022} \approx 0.0573$$
$$P1 = \frac{164.022}{9.9742 + 164.022} \approx 0.9427$$

- Decision: Spam (P1 $>$ 0.5)

Network Architecture

Core Features

- **Dual-output design**: Two neurons for two classes
- **Probability normalization**: $\sum P_i = 1$
- **Symmetric optimization**: Learns both decision boundaries

Mathematical Formulation

Softmax Function

For output scores $\mathbf{s} = [s_1, s_2]$:

$$P(y=1) = \frac{e^{s_1}}{e^{s_1} + e^{s_2}} \quad P(y=2) = \frac{e^{s_2}}{e^{s_1} + e^{s_2}}$$

Cross-Entropy Loss

$$L = -\sum_{c=1}^{2} y_c \log(p_c)$$

Where:

- y: One-hot encoded true label
- p: Predicted probability

Calculation Example: Correct Prediction

Parameters

- Output scores: [3.0, -1.0]
- True label: Class 0 (y=[1,0])

Calculation Example: Correct Prediction

Step-by-Step Calculation

1. Compute exponents:

$$e^{3.0} \approx 20.0855$$
$$e^{-1.0} \approx 0.3679$$

2. Compute probabilities:

$$P(y = 0) = 20.0855/(20.0855 + 0.3679) \approx 0.9820$$
$$P(y = 1) = 0.3679/(20.0855 + 0.3679) \approx 0.0180$$

3. Cross-entropy loss:

$$L = -[1 \times \log(0.9820) + 0 \times \log(0.0180)] \approx 0.0182$$

Calculation Example: Incorrect Prediction

Parameters

- Output scores: [-2.0, 1.0]
- True label: Class 0 (y=[1,0])

Step-by-Step Calculation

1. Compute exponents:

$$e^{-2.0} \approx 0.1353, e^{1.0} \approx 2.7183$$

$$P(y = 0) = 0.1353/(0.1353 + 2.7183) \approx 0.0474$$
$$P(y = 1) = 2.7183/(0.1353 + 2.7183) \approx 0.9526$$

2. Cross-entropy loss:

$$L = -[1 \times \log(0.0474) + 0 \times \log(0.9526)] \approx 3.0486$$

Sigmoid vs. Softmax Comparison

Characteristic	**Sigmoid**	**Softmax**
Output dimension	1	2
Loss calculation	$-y\log(p) - (1-y)\log(1-p)$	$-\sum y_c \log(p_c)$
Example 1 loss	0.018	0.018
Example 2 loss	3.05	3.05
Gradient calculation	$p - y$	$p_i - y_i$

Table: Comparison between Sigmoid and Softmax approaches

Application Scenarios

Medical

- Disease diagnosis
- Medical imaging

Industrial

- Quality control
- Predictive maintenance

Financial

- Fraud detection
- Risk assessment

NLP/CV

- Spam detection
- Object recognition

Recommendations

- Softmax: Better for multi-class extension
- Sigmoid: When single output suffices

Implementation Notes

Practical Considerations

- PyTorch's `CrossEntropyLoss` combines Softmax + NLL
- Numerical stability: Built-in log-softmax optimization
- Class imbalance: Use weighted loss

Training Tips

- Monitor both classes' accuracy
- Check gradient magnitudes
- Visualize decision boundaries