

Module 5: Multiclass Classification and Metrics

Overview

This module extends binary classification to scenarios with more than two classes and introduces comprehensive evaluation metrics. You will learn how softmax converts model outputs into probability distributions over multiple classes, how categorical cross-entropy measures classification performance, and how metrics like precision, recall, and confusion matrices provide nuanced evaluation beyond simple accuracy. The module also examines the fundamental limitations of linear decision boundaries.

1. From Binary to Multiclass

The Challenge

Binary classification distinguishes between two classes. But many problems have more: - Image recognition: 1000 categories - Digit recognition: 10 classes (0-9) - Disease diagnosis: Multiple conditions

Two Strategies

One-vs-All (OvA): - Train k binary classifiers (one per class) - Each classifier: "Is this class X or not?" - Predict the class with highest confidence

Softmax (Direct Multiclass): - Train a single model with k outputs - Softmax converts outputs to probabilities - Predict the class with highest probability

Modern deep learning typically uses softmax directly.

2. The Softmax Function

Definition

Softmax converts a vector of arbitrary real numbers (logits) into a probability distribution:

$$\text{softmax}(z)_i = \exp(z_i) / \sum_j \exp(z_j)$$

Properties

- **Output sums to 1:** Valid probability distribution
- **All outputs positive:** $\exp(z)$ is always positive
- **Preserves ordering:** Higher logit \Rightarrow higher probability
- **Sensitive to differences:** Amplifies gaps between values

Example

Logits: [2.0, 1.0, 0.1]

$\exp([2.0, 1.0, 0.1]) = [7.39, 2.72, 1.11]$
Sum = 11.22
Probabilities = [0.66, 0.24, 0.10]

The model is 66% confident in class 0.

Relationship to Sigmoid

Softmax with 2 classes reduces to sigmoid: - $\text{softmax}([z, 0]) = [\text{sigmoid}(z), 1 - \text{sigmoid}(z)]$

3. Categorical Cross-Entropy Loss

Definition

For a single example with true class c :

$$L = -\log(p_c)$$

Where p_c is the predicted probability of the true class.

For a batch with one-hot encoded labels:

$$L = -(1/n) \sum_i \sum_k y_{ik} * \log(p_{ik})$$

Intuition

- If the true class is c and we predict $p_c = 0.99$, loss = $-\log(0.99) \approx 0.01$ (low)
- If the true class is c and we predict $p_c = 0.01$, loss = $-\log(0.01) \approx 4.6$ (high)

The loss heavily penalizes confident wrong predictions.

Gradient

The gradient has a beautifully simple form:

$$L / z_k = p_k - y_k$$

For the true class: gradient = (predicted probability - 1) For other classes: gradient = (predicted probability - 0)

4. Precision and Recall

Why Accuracy Isn't Enough

Consider fraud detection with 1% fraud rate: - A model predicting "not fraud" always achieves 99% accuracy - But it catches zero fraudsters!

We need metrics that handle class imbalance.

Precision

Of the examples we predicted positive, how many are actually positive?

$\text{Precision} = \text{True Positives} / (\text{True Positives} + \text{False Positives})$

High precision means: "When I say positive, I'm usually right."

Recall

Of the examples that are actually positive, how many did we predict correctly?

$\text{Recall} = \text{True Positives} / (\text{True Positives} + \text{False Negatives})$

High recall means: "I find most of the positives."

The Precision-Recall Tradeoff

- **High threshold:** High precision (conservative), low recall (miss many)
- **Low threshold:** Low precision (many false alarms), high recall (find most)

The optimal tradeoff depends on the application.

5. F1 Score

Harmonic Mean

F1 score balances precision and recall:

$F1 = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$

Why Harmonic Mean?

The harmonic mean is low if either precision or recall is low: - $P=0.9, R=0.9 \Rightarrow F1=0.90$ - $P=0.9, R=0.1 \Rightarrow F1=0.18$

This penalizes imbalance between precision and recall.

When to Use

F1 is appropriate when: - Both precision and recall matter - Classes are imbalanced - You need a single number for comparison

6. Confusion Matrix

Structure

A confusion matrix shows all predictions vs. actual classes:

		Predicted	
		Neg	Pos
Actual	Neg	TN	FP
	Pos	FN	TP

- **TN (True Negative)**: Correctly predicted negative
- **FP (False Positive)**: Incorrectly predicted positive (Type I error)
- **FN (False Negative)**: Incorrectly predicted negative (Type II error)
- **TP (True Positive)**: Correctly predicted positive

Multiclass Extension

For k classes, the confusion matrix is k×k: - Row = actual class - Column = predicted class - Diagonal = correct predictions - Off-diagonal = errors (which mistakes are being made)

Derived Metrics

From the confusion matrix: - Accuracy = $(TP + TN) / (TP + TN + FP + FN)$ - Precision = $TP / (TP + FP)$ - Recall = $TP / (TP + FN)$ - Specificity = $TN / (TN + FP)$

7. ROC and AUC

ROC Curve

The Receiver Operating Characteristic curve plots: - Y-axis: True Positive Rate (Recall) - X-axis: False Positive Rate (1 - Specificity)

Each point on the curve represents a different threshold.

Area Under Curve (AUC)

- **AUC = 1.0**: Perfect classifier
- **AUC = 0.5**: Random guessing (diagonal line)
- **AUC < 0.5**: Worse than random

AUC measures discrimination ability across all thresholds.

When to Use

ROC/AUC is useful for: - Comparing classifiers independent of threshold - Balanced datasets - When false positive costs equal false negative costs

8. Limitations of Linear Models

Linear Decision Boundaries

Logistic regression (and softmax regression) can only create linear decision boundaries. In 2D, these are straight lines; in general, hyperplanes.

The XOR Problem

Consider data where: - (0,0) □ Class 0 - (1,1) □ Class 0 - (0,1) □ Class 1 - (1,0) □ Class 1

No straight line can separate these! This is the classic XOR problem.

When Linear Fails

Linear models struggle when: - True boundaries are curved - Classes are interleaved - Features interact in complex ways

Solutions

1. **Feature engineering**: Add polynomial or crossed features
 2. **Neural networks**: Learn non-linear boundaries (Module 6)
 3. **Kernel methods**: Implicit high-dimensional features
 4. **Tree-based models**: Non-linear by construction (Module 7)
-

Key Takeaways

1. **Softmax** generalizes sigmoid to multiple classes
 2. **Cross-entropy loss** is the standard for classification
 3. **Accuracy is insufficient** for imbalanced problems
 4. **Precision**: How reliable are positive predictions?
 5. **Recall**: How complete are positive predictions?
 6. **F1 score** balances precision and recall
 7. **Confusion matrix** shows all prediction outcomes
 8. **Linear models have limits**—motivating neural networks
-

Connections to Future Modules

- **Module 6**: Neural networks for non-linear boundaries
 - **Module 7**: Tree-based models (inherently non-linear)
 - **Module 12**: Fairness metrics (subgroup precision/recall)
-

Further Reading

- [Links to be added]
- Introduction to Statistical Learning, Chapter 4
- scikit-learn documentation on metrics