Lecture 3: Logistic Regression

Luu Minh Sao Khue

From probability to prediction — sigmoid function, log-loss, decision boundary, odds interpretation, and evaluation metrics (precision, recall, F1, ROC-AUC). Includes implementation and analysis using `scikit-learn`.

1. Learning Objectives

After this lecture, you will be able to:

- Explain why linear regression is unsuitable for classification.
- Derive the logistic (sigmoid) model from probability principles.
- Interpret coefficients in terms of odds and log-odds.
- Derive the log-loss and maximum likelihood formulation.
- Extend logistic regression to multiclass (one-vs-rest).
- Evaluate classification models using key metrics: precision, recall, F1-score, ROC-AUC, and confusion matrix.
- Implement and interpret logistic regression in Python.

2. Motivation and Intuition

Linear regression predicts continuous outcomes, but classification requires probabilities between 0 and 1. If we directly apply a linear model, predicted values can fall outside $[0, 1]$. Logistic regression solves this by transforming the linear combination of inputs through the **sigmoid function**, producing a valid probability estimate.

Example: Predict if a patient has a disease ($y = 1$) or not ($y = 0$) based on biomarkers $x_1, x_2, \ldots, x_p$.

We want:

$$P(y = 1|x) = \sigma(z) = \frac{1}{1 + e^{-z}}, \quad z = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p.$$

3. Logistic (Sigmoid) Function and Decision Boundary

3.1 Sigmoid Function

$$\sigma(z) = \frac{1}{1+e^{-z}}, \quad \sigma'(z) = \sigma(z)[1-\sigma(z)].$$

- $\sigma(z) \in (0,1)$ maps any real number to a probability.
- As $z \to +\infty$, $\sigma(z) \to 1$; as $z \to -\infty$, $\sigma(z) \to 0$.
- Symmetric about $z = 0$ where $\sigma(0) = 0.5$.

3.2 Decision Boundary

Prediction rule:

$$\hat{y} = \begin{cases} 1, & \text{if } P(y=1|x) \geq 0.5 \\ 0, & \text{otherwise.} \end{cases}$$

Equivalently,

$$\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p = 0$$

defines the **decision boundary**, separating the classes in feature space.

4. Odds and Log-Odds Interpretation

The model assumes a linear relationship between predictors and the **log-odds** (logit):

$$\text{logit}(P(y=1|x)) = \log \frac{P(y=1|x)}{1-P(y=1|x)} = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p.$$

- The **odds** of success are $\frac{P(y=1|x)}{1-P(y=1|x)}$.
- Increasing x_j by one unit multiplies the odds by e^{β_j}.

Interpretation.

- $\beta_j > 0$ increases odds (positive association).
- $\beta_j < 0$ decreases odds.
- Intercept β_0 gives log-odds when all $x_j = 0$.

5. Log-Loss and Maximum Likelihood Estimation

Given data $\{(x_i, y_i)\}_{i=1}^n$, where $y_i \in \{0, 1\}$, the model predicts

$$p_i = P(y_i = 1|x_i) = \sigma(x_i^T \beta).$$

The likelihood is

$$L(\beta) = \prod_{i=1}^{n} p_i^{y_i} (1 - p_i)^{1-y_i}.$$

Log-likelihood:

$$\ell(\beta) = \sum_{i=1}^{n} \big[y_i \log p_i + (1 - y_i) \log(1 - p_i) \big].$$

We maximize $\ell(\beta)$ (equivalently minimize the negative):

$$J(\beta) = -\frac{1}{n} \sum_{i=1}^{n} \big[y_i \log p_i + (1 - y_i) \log(1 - p_i) \big],$$

known as the **log-loss** or **binary cross-entropy**.

5.1 Gradient and Optimization

$$\nabla_\beta J(\beta) = \frac{1}{n} X^T (\sigma(X\beta) - y).$$

Optimization is performed via gradient descent or variants (e.g. Newton–Raphson). No closed form exists as in linear regression.

6. Multiclass Logistic Regression (One-vs-Rest)

For $K > 2$ classes, train K separate binary classifiers:

$$P(y = k|x) = \frac{1}{1 + \exp(-(\beta_{0k} + \beta_k^T x))}, \quad k = 1, \ldots, K.$$

Each model predicts class k vs. all others. The final label is

$$\hat{y} = \arg\max_k P(y = k|x).$$

Alternative: the **softmax regression** model uses

$$P(y = k|x) = \frac{\exp(\beta_k^T x)}{\sum_{j=1}^{K} \exp(\beta_j^T x)},$$

with the cross-entropy loss generalized accordingly.

7. Model Evaluation Metrics

7.1 Confusion Matrix

For binary classification:

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}.$$

7.2 Precision, Recall, and F1

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN},$$
$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

- Precision: reliability of positive predictions.
- Recall: coverage of actual positives.
- F1: harmonic mean, balances both.

7.3 ROC Curve and AUC

- **ROC curve:** plots True Positive Rate (TPR) vs. False Positive Rate (FPR) for all thresholds.
- **AUC:** area under the ROC curve, measures ranking quality.

8. Python Implementation

```
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (
    confusion_matrix, precision_score, recall_score,
    f1_score, roc_auc_score, RocCurveDisplay
)
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_classification
import matplotlib.pyplot as plt

# Generate synthetic binary data
X, y = make_classification(
    n_samples=200, n_features=3, n_informative=2,
    n_redundant=0, random_state=42
)
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# Fit logistic regression
model = LogisticRegression()
model.fit(X_train, y_train)

# Predict and evaluate
y_pred = model.predict(X_test)
y_prob = model.predict_proba(X_test)[:, 1]

print("Precision:", precision_score(y_test, y_pred))
print("Recall:", recall_score(y_test, y_pred))
print("F1:", f1_score(y_test, y_pred))
print("ROC-AUC:", roc_auc_score(y_test, y_prob))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

# Plot ROC curve
RocCurveDisplay.from_estimator(model, X_test, y_test)
plt.show()
```

9. Interpretation and Practical Tips

- Coefficient β_j: change in *log-odds* per unit change in x_j.
- e^{β_j} gives the multiplicative effect on odds.
- Standardize features before training for stable optimization.
- Use regularization (`penalty='l2'`) to prevent overfitting.
- For imbalanced data, inspect Precision–Recall curve instead of ROC.
- Choose classification threshold based on the application's cost trade-off.

10. Summary

- Logistic regression models $P(y = 1|x) = \sigma(x^T\beta)$.
- Log-odds (logit) are linear in predictors.
- Parameters estimated by maximizing likelihood $\Rightarrow$ minimizing log-loss.
- Decision boundary: $\beta_0 + \beta^T x = 0$.
- Multiclass: one-vs-rest or softmax regression.
- Evaluation metrics: Precision, Recall, F1, ROC-AUC, Confusion Matrix.
- Practice: normalize features, use regularization, adjust threshold for desired trade-offs.

11. Exercises

1. Derive the gradient of the log-loss for logistic regression.
2. Show that $\sigma(-z) = 1 - \sigma(z)$.
3. Implement logistic regression using gradient descent from scratch.
4. Compare one-vs-rest and softmax regression on a multiclass dataset.
5. Compute Precision, Recall, F1, and ROC-AUC for varying thresholds.
6. Explain how changing the threshold affects F1 and ROC-AUC.