Lecture 13: Classification and Logistic Regression

CS109A: Introduction to Data Science

Harvard University

Course:	CS109A: Introduction to Data Science
Lecture:	Lecture 13
Instructors:	Pavlos Protopapas, Kevin Rader, Chris Gumb
Topics:	Classification vs. Regression, Why linear regression fails for classification, Logistic Regression, Sigmoid function, Odds and Log-Odds interpretation, Maximum Likelihood Estimation, Binary Cross-Entropy, Decision Boundaries

Contents

1 Introduction: From Regression to Classification

Lecture Overview

This lecture marks a major transition in the course: we move from **regression** problems (predicting numeric values) to **classification** problems (predicting categories).

Key Learning Objectives:

- Understand the fundamental difference between regression and classification
- Learn why linear regression is unsuitable for classification tasks
- Master logistic regression: the foundational parametric model for classification
- Interpret logistic regression coefficients in terms of odds and log-odds
- Understand the probabilistic foundation: Bernoulli likelihood and MLE
- Visualize and understand decision boundaries

This lecture also includes a review section covering hypothesis testing, permutation tests, and interaction terms for midterm preparation.

1.1 Regression vs. Classification

Definition: Regression Problems

In **regression**, the response variable Y is **quantitative** (numeric, continuous).

Examples:

- Predicting house prices ($500,000)
- Predicting tomorrow's temperature (72.5°F)
- Predicting sales revenue ($1.2 million)

Models: Linear regression, polynomial regression, Ridge, Lasso, k-NN (for regression)

Definition: Classification Problems

In **classification**, the response variable Y is **qualitative** (categorical).

Examples:

- Predicting whether an email is spam or not spam (binary)
- Predicting whether a patient has heart disease (Yes/No)
- Predicting which major a student will choose (CS/Stats/Other)
- Predicting handwritten digits (0-9)

Models: Logistic regression, k-NN (for classification), decision trees, neural networks

Example: Regression vs. Classification Question

Consider a medical dataset with patient information:

Regression question: "What will this patient's maximum heart rate be?"

- Answer: A number (e.g., 152 bpm)

Classification question: "Does this patient have heart disease?"

- Answer: A category (Yes or No)

2 Why Not Linear Regression for Classification?

A natural first thought might be: "Can't we just use linear regression for classification by encoding categories as numbers?" Let's see why this fails.

2.1 Problem 1: Multi-class Encoding Creates False Ordering

Suppose we want to predict a student's major with three categories: CS, Statistics, and Other. We might encode these as:

- $Y = 1$ if CS
- $Y = 2$ if Statistics
- $Y = 3$ if Other

Important Note

The Problem: Linear regression assumes numerical relationships between Y values!
When we fit a linear regression, the model interprets:

- The "distance" from CS (1) to Statistics (2) is the same as from Statistics (2) to Other (3)
- A one-unit change in X corresponds to moving one "step" on this scale

This is **completely meaningless** for categorical data! The categories have no inherent numerical order or spacing.
If we had encoded differently ($Y = 1$ for Other, $Y = 2$ for CS, $Y = 3$ for Statistics), we'd get a completely different model!

2.2 Problem 2: Binary Predictions Can Exceed [0, 1]

Even with binary classification (only two categories), linear regression fails in a different way.

Consider predicting heart disease (Yes = 1, No = 0) based on maximum heart rate. If we fit a linear regression $Y = \beta_0 + \beta_1 X$, we can interpret $\hat{Y}$ as the "probability" that $Y = 1$.

Important Note

The Problem: Probabilities must be between 0 and 1!
A straight line has no bounds. For extreme values of X:

- **Very low heart rate:** Model might predict $P(\text{heart disease}) = 1.1$ (110%)
- **Very high heart rate:** Model might predict $P(\text{heart disease}) = -0.2$ (-20%)

These predictions are mathematically and logically invalid!

Example: Linear Regression for Heart Disease

If we fit a linear regression to predict heart disease from maximum heart rate, we get:

- A downward sloping line (higher heart rate $\rightarrow$ lower probability)
- But for patients with very low heart rates (e.g., 50 bpm), the model predicts probabilities > 1
- For very fit athletes with high heart rates (e.g., 200 bpm), the model predicts negative probabilities

We need a function that "squashes" predictions to stay between 0 and 1!

3 Logistic Regression: The Solution

3.1 The Sigmoid Function

Instead of predicting probabilities directly with a line, we use an **S-shaped curve** that is naturally bounded between 0 and 1.

Definition: Sigmoid (Logistic) Function

The **sigmoid function** maps any real number to a value between 0 and 1:

$$\sigma(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{1+e^z} \tag{1}$$

Properties:

- As $z \to +\infty$: $\sigma(z) \to 1$
- As $z \to -\infty$: $\sigma(z) \to 0$
- At $z = 0$: $\sigma(0) = 0.5$
- The function is **monotonically increasing**
- Output is always in $(0, 1)$

3.2 The Logistic Regression Model

In logistic regression, we model the **probability** that $Y = 1$:

$$P(Y = 1|X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X)}} \tag{2}$$

Or equivalently:

$$P(Y = 1|X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}} \tag{3}$$

Key Summary

Logistic Regression in Two Steps:

1. **Linear part (same as linear regression):** Compute $h = \beta_0 + \beta_1 X$
2. **Sigmoid transformation:** Pass h through the sigmoid to get probability $p = \sigma(h)$

The sigmoid "squashes" the linear prediction to always be between 0 and 1.

3.3 Understanding the Shape

The parameters β_0 and β_1 control the shape of the S-curve:

- β_0 (intercept): Controls the **horizontal position** of the curve
  - Larger β_0 shifts the curve to the left (higher probabilities for the same X)
  - Smaller β_0 shifts the curve to the right
- β_1 (slope): Controls the **steepness** of the curve

- Larger $|\beta_1|$ makes the transition from 0 to 1 sharper
- Smaller $|\beta_1|$ makes the transition more gradual
- Positive β_1: Probability increases as X increases
- Negative β_1: Probability decreases as X increases

4 Interpreting Logistic Regression: Odds and Log-Odds

4.1 The Problem with Direct Interpretation

With linear regression, interpretation is simple: "A one-unit increase in X is associated with a β_1 change in Y."

With logistic regression, the relationship between X and $P(Y = 1)$ is non-linear (S-shaped), so this simple interpretation doesn't work.

4.2 Odds

Definition: Odds

The **odds** of an event is the ratio of the probability of success to the probability of failure:

$$\text{Odds} = \frac{p}{1-p} \tag{4}$$

where $p = P(\text{success})$.

Examples:

- If $p = 0.5$ (50% chance): Odds $= \frac{0.5}{0.5} = 1$ (1:1, or "even odds")
- If $p = 0.8$ (80% chance): Odds $= \frac{0.8}{0.2} = 4$ (4:1, "4 to 1")
- If $p = 0.2$ (20% chance): Odds $= \frac{0.2}{0.8} = 0.25$ (1:4)

Intuition: Odds tell you "how many times more likely is success than failure?"

4.3 Log-Odds (Logit)

Definition: Log-Odds / Logit

The **log-odds** (also called **logit**) is the natural logarithm of the odds:

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right) \tag{5}$$

Key properties:

- Log-odds can be any real number ($-\infty$ to $+\infty$)
- When $p = 0.5$: log-odds $= \ln(1) = 0$
- When $p > 0.5$: log-odds > 0
- When $p < 0.5$: log-odds < 0

4.4 The Log-Odds Formulation

Here's the key insight: if we solve the logistic regression equation for the linear part, we get:

$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X \tag{6}$$

Key Summary

Logistic Regression Models Log-Odds Linearly!
The log-odds of success is a **linear function** of the predictors. This is why logistic regression is still considered a "linear" model.

4.5 Interpreting Coefficients

Key Information

β_1 Interpretation (Log-Odds):
A one-unit increase in X is associated with a β_1 **additive change** in the log-odds of $Y = 1$.
e^{β_1} Interpretation (Odds Ratio):
A one-unit increase in X is associated with the odds being **multiplied by** e^{β_1}.

Example: Heart Disease Model

Suppose we fit a logistic regression to predict heart disease from maximum heart rate and get:

$$\ln\left(\frac{P(\text{HeartDisease})}{1-P(\text{HeartDisease})}\right) = 6.325 - 0.0434 \times \text{MaxHR} \tag{7}$$

Interpreting $\beta_1 = -0.0434$:
Log-odds interpretation:

- For each 1 bpm increase in maximum heart rate, the log-odds of heart disease **decrease** by 0.0434.

Odds ratio interpretation:

- $e^{-0.0434} \approx 0.957$
- For each 1 bpm increase in maximum heart rate, the odds of heart disease are **multiplied by 0.957** (i.e., decrease by about 4.3%).

Intuition: Higher maximum heart rate is associated with lower risk of heart disease.

Important Note

Special Values of β_1:

- $\beta_1 = 0$: $e^0 = 1$ (odds multiplied by 1 = no change) $\rightarrow$ **No association**
- $\beta_1 > 0$: $e^{\beta_1} > 1$ (odds increase) $\rightarrow$ **Positive association**
- $\beta_1 < 0$: $e^{\beta_1} < 1$ (odds decrease) $\rightarrow$ **Negative association**

5 Estimating Logistic Regression: MLE

5.1 The Probabilistic Perspective

In linear regression, we assumed errors were normally distributed and minimized MSE (equivalent to maximizing Gaussian likelihood).

In logistic regression, our outcomes are binary (0 or 1), so we use a different distribution.

Definition: Bernoulli Distribution

A **Bernoulli random variable** Y takes value 1 with probability p and value 0 with probability $1 - p$:

$$P(Y = y) = p^y (1 - p)^{1-y} \tag{8}$$

This is like a single coin flip with probability p of heads.

5.2 Likelihood Function

For logistic regression, we assume each observation y_i comes from a Bernoulli distribution with probability p_i that depends on X_i:

$$p_i = P(Y_i = 1|X_i) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_i)}} \tag{9}$$

The **likelihood** of observing our entire dataset is:

$$L(\beta) = \prod_{i=1}^{n} p_i^{y_i} (1 - p_i)^{1-y_i} \tag{10}$$

Intuition:

- If $y_i = 1$ (actual success), we want p_i to be high
- If $y_i = 0$ (actual failure), we want $1 - p_i$ to be high
- Good parameters make observed successes have high predicted probabilities

5.3 Maximum Likelihood Estimation

Definition: Maximum Likelihood Estimation (MLE)

MLE finds the parameter values $\hat{\beta}$ that **maximize** the likelihood of observing the data we actually observed:

$$\hat{\beta}_{MLE} = \arg\max_{\beta} L(\beta) \tag{11}$$

5.4 Log-Likelihood and Binary Cross-Entropy

Products are hard to work with, so we take the logarithm:

$$\ell(\beta) = \log L(\beta) = \sum_{i=1}^{n} [y_i \log(p_i) + (1 - y_i)\log(1 - p_i)] \tag{12}$$

Since computers typically **minimize** functions, we minimize the **negative** log-likelihood:

Definition: Binary Cross-Entropy Loss

The **binary cross-entropy** (BCE) loss is the negative log-likelihood:

$$\text{BCE} = -\sum_{i=1}^{n} [y_i \log(p_i) + (1 - y_i)\log(1 - p_i)] \tag{13}$$

This is the loss function that logistic regression minimizes.

Key Summary

Comparison: Linear vs. Logistic Regression

	Linear Regression	**Logistic Regression**
Response type	Continuous	Binary (0/1)
Distribution assumption	Normal	Bernoulli
Loss function	MSE	Binary Cross-Entropy
Estimation method	Closed-form OLS	Iterative optimization (MLE)

Important Note

Why No σ^2 Parameter?
In linear regression, we estimate β_0, β_1, and σ^2 (error variance).
In logistic regression, we only estimate β_0 and β_1. Why?
Answer: The variance is determined by the Bernoulli distribution itself! If $P(Y = 1) = p$, then $\text{Var}(Y) = p(1-p)$. There's no separate "noise" parameter—the randomness comes from the binary nature of the outcome.

6 Multiple Logistic Regression and Decision Boundaries

6.1 Multiple Logistic Regression

Just like linear regression extends to multiple predictors, so does logistic regression:

$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p \tag{14}$$

Interpretation: e^{β_j} is the multiplicative change in odds for a one-unit increase in X_j, **holding all other predictors constant**.

6.2 Making Predictions: From Probability to Class

Logistic regression outputs a **probability**. To make a classification decision, we need a **threshold**:

- Default threshold: $t = 0.5$
- If $P(Y = 1|X) \geq t$: Predict class 1
- If $P(Y = 1|X) < t$: Predict class 0

6.3 Decision Boundaries

Definition: Decision Boundary

The **decision boundary** is the set of points where the model is exactly undecided—where $P(Y = 1) = 0.5$.
For logistic regression, $P(Y = 1) = 0.5$ when the log-odds equal 0:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0 \tag{15}$$

6.4 Linear vs. Non-linear Boundaries

Linear boundary (default):

With basic logistic regression, the decision boundary is a **linear equation** in the features. In 2D (X_1, X_2), this is a **straight line**.

Non-linear boundary:

To create curved decision boundaries, add **polynomial** or **interaction** terms:

$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \beta_4 X_1^2 + \beta_5 X_2^2 \tag{16}$$

Setting this equal to 0 gives a **quadratic equation** in X_1 and X_2, which can produce **circles, ellipses, or hyperbolas**.

Important Note

The Model is Still "Linear"!

Even with polynomial features, we still call this a "linear model" because it's linear in the **parameters** (β's).

The trick is that we create non-linear **features** (X_1^2, X_1X_2, etc.) and then do linear regression on those features.

7 Review Section: Hypothesis Testing and Permutation Tests

This section reviews key concepts for the midterm.

7.1 Hypothesis Testing Framework

The Five Steps:

1. **State hypotheses:**
   - H_0: No effect ($\beta_1 = 0$)
   - H_A: There is an effect ($\beta_1 \neq 0$)
2. **Choose test statistic:** Usually $t = \hat{\beta}_1 / SE(\hat{\beta}_1)$
3. **Compute test statistic** from data
4. **Calculate p-value:** Probability of seeing a result this extreme if H_0 is true
5. **Make decision:** Reject H_0 if p-value $< \alpha$ (usually 0.05)

7.2 Permutation Tests

When t-test assumptions (normality, constant variance) are violated:

1. Compute the observed test statistic (e.g., $\hat{\beta}_1 = 0.589$)
2. **Shuffle** Y randomly while keeping X fixed (simulates H_0: no relationship)
3. Fit model to shuffled data, record $\hat{\beta}_1^*$
4. Repeat 1000+ times to build the **null distribution**
5. P-value = proportion of permuted statistics as extreme as observed

7.3 Bootstrap vs. Permutation

	Bootstrap	**Permutation Test**
Purpose	Estimation (confidence intervals)	Hypothesis testing (p-values)
Sampling	With replacement (pairs)	Without replacement (shuffle Y)
Assumption	Data represents population	H_0 is true

7.4 Interaction Terms

In a model like `price ~ sqft * type`:

- **Main effect for sqft:** Effect of sqft for the **reference category**
- **Interaction term (sqft:type):** How the sqft effect **differs** for other categories compared to reference
- **Total effect** for category k: Main effect + interaction term for k

7.5 Confidence vs. Prediction Intervals

- **Confidence interval:** Where we think the **mean** response lies (narrower)
- **Prediction interval:** Where a **new individual** response would lie (wider, includes ϵ)

8 Implementation in Python

8.1 Logistic Regression with sklearn

```python
from sklearn.linear_model import LogisticRegression

# Prepare data
X = df[['MaxHR']]  # Must be 2D array
y = df['HeartDisease']  # Binary: 0 or 1

# Fit logistic regression (no regularization)
logreg = LogisticRegression(penalty=None)  # Note: use None, not 'none'
logreg.fit(X, y)

# Get coefficients
print("beta_1 (coefficient):", logreg.coef_[0][0])
print("beta_0 (intercept):", logreg.intercept_[0])

# Make predictions
probabilities = logreg.predict_proba(X)  # P(Y=0) and P(Y=1)
predictions = logreg.predict(X)  # Class labels (0 or 1)
```

8.2 Permutation Test Implementation

```python
import numpy as np
from sklearn.linear_model import LinearRegression

# Observed statistic
model = LinearRegression().fit(X, y)
observed_beta = model.coef_[0]

# Permutation test
n_perms = 1000
perm_betas = []

for _ in range(n_perms):
    # Shuffle Y (breaks any real relationship)
    y_shuffled = np.random.permutation(y)

    # Fit model to shuffled data
    model_perm = LinearRegression().fit(X, y_shuffled)
    perm_betas.append(model_perm.coef_[0])

# P-value: proportion of permuted betas as extreme as observed
p_value = np.mean(np.abs(perm_betas) >= np.abs(observed_beta))
```

9 Key Takeaways

Key Summary

Why Classification Needs Different Methods:

- Linear regression for multi-class creates meaningless numerical ordering
- Linear regression for binary can predict probabilities outside [0, 1]

Logistic Regression:

- Uses sigmoid function to squash predictions to (0, 1)
- Models log-odds as a linear function of features
- Coefficient interpretation: e^{β_1} = multiplicative change in odds

Estimation:

- Assumes Bernoulli distribution for binary outcomes
- Maximizes likelihood (minimizes binary cross-entropy)
- No σ^2 parameter—variance determined by $p(1-p)$

Decision Boundaries:

- Linear model → linear boundary
- Add polynomial/interaction features → curved boundary
- Model is still "linear" in parameters

10 Quick Reference

Key Information

Logistic Regression Model:

$$P(Y=1) = \frac{1}{1+e^{-(\beta_0+\beta_1 X)}} \tag{17}$$

Log-Odds Form:

$$\ln\left(\frac{P(Y=1)}{1-P(Y=1)}\right) = \beta_0 + \beta_1 X \tag{18}$$

Odds Ratio:

$$\text{Odds} = \frac{p}{1-p}, \quad \text{OR} = e^{\beta_1} \tag{19}$$

Binary Cross-Entropy Loss:

$$\text{BCE} = -\sum_{i=1}^{n} [y_i \log(p_i) + (1-y_i)\log(1-p_i)] \tag{20}$$

Decision Boundary (threshold = 0.5):

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots = 0 \tag{21}$$