Detailed Logistic Regression Analysis Examples

Example 1: Simple Hand Calculation

Let's start with a minimal example that we can solve by hand.

Problem Setup

We have 3 observations of a treatment at different doses:

- Dose 0: 0 successes out of 10 trials
- Dose 1: 3 successes out of 10 trials
- Dose 2: 8 successes out of 10 trials

Step 1: Set up the model

Model: $\theta(x) = \sigma(\alpha + \beta x)$ where $\sigma(z) = 1/(1 + e^{(-z)})$

Data:

- x = [0, 1, 2] (doses)
- y = [0, 3, 8] (successes)
- N = 10 (trials per dose)

Step 2: Calculate log-likelihood

Log-likelihood: $\log L(\alpha, \beta) = \Sigma[y_i \log(\theta_i) + (N_i - y_i)\log(1 - \theta_i)]$

For $\alpha = -2$, $\beta = 2$:

- $x_0 = 0$: $\theta_0 = \sigma(-2) = 1/(1 + e^2) \approx 0.119$
  - $\log L_0 = 0\times\log(0.119) + 10\times\log(0.881) \approx -1.27$
- $x_1 = 1$: $\theta_1 = \sigma(-2 + 2) = \sigma(0) = 0.5$
  - $\log L_1 = 3\times\log(0.5) + 7\times\log(0.5) = 10\times\log(0.5) \approx -6.93$
- $x_2 = 2$: $\theta_2 = \sigma(-2 + 4) = \sigma(2) \approx 0.881$
  - $\log L_2 = 8\times\log(0.881) + 2\times\log(0.119) \approx -5.21$

Total log-likelihood ≈ -13.41

Step 3: Grid search (simplified)

Let's evaluate on a small grid:

α\β	1.5	2.0	2.5
-2.5	-15.2	-14.1	-13.8
-2.0	-14.5	-13.4	-13.5
-1.5	-14.2	-13.2	-13.7

Best values: $\alpha \approx -2.0$, $\beta \approx 2.0$

Step 4: Predictions

For a new dose $x^* = 1.5$: $\theta^* = \sigma(-2 + 2\times1.5) = \sigma(1) \approx 0.731$

Expected successes: $10 \times 0.731 \approx 7.3$

Example 2: Complete Analysis with Code

python

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import binom
from scipy.optimize import minimize

# Data: Drug effectiveness study
doses = np.array([0, 0.5, 1.0, 1.5, 2.0, 2.5])
trials = np.array([20, 20, 20, 20, 20, 20])
successes = np.array([1, 3, 8, 14, 18, 19])

# Step 1: Standardize covariates
x_mean = doses.mean()
x_std = doses.std()
x = (doses - x_mean) / x_std

# Step 2: Define the model
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

def log_likelihood(params, x, y, n):
    alpha, beta = params
    theta = sigmoid(alpha + beta * x)
    # Avoid log(0) issues
    theta = np.clip(theta, 1e-10, 1-1e-10)
    return np.sum(y * np.log(theta) + (n - y) * np.log(1 - theta))

def neg_log_likelihood(params, x, y, n):
    return -log_likelihood(params, x, y, n)

# Step 3: Find MLE using optimization
initial_guess = [0, 0]
result = minimize(neg_log_likelihood, initial_guess,
                  args=(x, successes, trials))
alpha_mle, beta_mle = result.x

print(f"MLE estimates: α = {alpha_mle:.3f}, β = {beta_mle:.3f}")

# Step 4: Make predictions
new_doses = np.linspace(0, 3, 100)
new_x = (new_doses - x_mean) / x_std
theta_pred = sigmoid(alpha_mle + beta_mle * new_x)

# Step 5: Visualization
```

```python
plt.figure(figsize=(10, 6))
plt.scatter(doses, successes/trials, s=100, c='red',
            label='Observed', zorder=3)
plt.plot(new_doses, theta_pred, 'b-', linewidth=2,
         label='Fitted curve')
plt.xlabel('Dose')
plt.ylabel('Success Rate')
plt.title('Logistic Regression: Dose-Response Curve')
plt.legend()
plt.grid(True, alpha=0.3)
plt.show()

# Step 6: Interpret coefficients
odds_ratio = np.exp(beta_mle * x_std)  # Per unit dose
print(f"\nOdds ratio per unit dose: {odds_ratio:.3f}")
print(f"This means each unit increase in dose multiplies "
      f"the odds of success by {odds_ratio:.3f}")
```

Example 3: Hand-Solved Bayesian Analysis

Problem Setup

Simple case with 2 observations:

- Temperature -1: 2 failures out of 3 trials
- Temperature +1: 0 failures out of 3 trials

Prior: $\alpha \sim N(0,1)$, $\beta \sim N(0,1)$

Step 1: Set up grid

Let's use a 3×3 grid:

- $\alpha \in \{-1, 0, 1\}$
- $\beta \in \{-1, 0, 1\}$

Step 2: Calculate for each grid point

For ($\alpha=0$, $\beta=-1$):

- $x_1=-1$: $\theta_1 = \sigma(0-1\times(-1)) = \sigma(1) \approx 0.731$
  - $p(y_1=2|\theta_1) = C(3,2) \times 0.731^2 \times 0.269^1 \approx 0.431$
- $x_2=+1$: $\theta_2 = \sigma(0-1\times(1)) = \sigma(-1) \approx 0.269$
  - $p(y_2=0|\theta_2) = C(3,0) \times 0.269^0 \times 0.731^3 \approx 0.391$

Likelihood: 0.431 × 0.391 ≈ 0.169 Prior: φ(0,0,1) × φ(-1,0,1) ≈ 0.399 × 0.242 ≈ 0.097 Joint: 0.169 × 0.097 ≈ 0.016

Step 3: Complete grid

	β=-1	β=0	β=1
α=-1	0.002	0.006	0.001
α=0	0.016	0.012	0.002
α=1	0.003	0.001	0.000

Step 4: Normalize

Z = Σ(all values) ≈ 0.043 Posterior probabilities = joint/Z

	β=-1	β=0	β=1
α=-1	0.047	0.140	0.023
α=0	0.372	0.279	0.047
α=1	0.070	0.023	0.000

Step 5: Posterior statistics

E[α] ≈ 0×0.698 + (-1)×0.210 + 1×0.093 ≈ -0.117 E[β] ≈ (-1)×0.489 + 0×0.442 + 1×0.070 ≈ -0.419

Example 4: Real-World Application

Medical Trial Analysis

python

```python
# Clinical trial data: Drug vs Placebo
# Outcome: Patient improved (1) or not (0)
# Covariate: Age (standardized)

import pandas as pd

# Create example data
data = pd.DataFrame({
    'age': [25, 30, 35, 40, 45, 50, 55, 60, 65, 70],
    'treatment': [1, 0, 1, 0, 1, 0, 1, 0, 1, 0],  # 1=drug, 0=placebo
    'improved': [1, 0, 1, 0, 1, 1, 0, 0, 0, 0],
    'n_patients': [10, 12, 15, 11, 13, 14, 16, 15, 12, 10]
})

# Standardize age
data['age_std'] = (data['age'] - data['age'].mean()) / data['age'].std()

# Model with interaction: logit(p) = α + β₁×age + β₂×treatment + β₃×age×treatment
X = np.column_stack([
    np.ones(len(data)),  # intercept
    data['age_std'],
    data['treatment'],
    data['age_std'] * data['treatment']
])

# Fit model (using simple gradient descent)
def fit_logistic_regression(X, y, n, learning_rate=0.01, iterations=1000):
    m, n_features = X.shape
    weights = np.zeros(n_features)
    
    for i in range(iterations):
        z = X @ weights
        predictions = sigmoid(z)
        gradient = X.T @ (predictions - y/n) / m
        weights -= learning_rate * gradient
    
    return weights

weights = fit_logistic_regression(X, data['improved'], data['n_patients'])
print("Model coefficients:")
print(f"Intercept (α): {weights[0]:.3f}")
print(f"Age effect (β₁): {weights[1]:.3f}")
print(f"Treatment effect (β₂): {weights[2]:.3f}")
```

```python
print(f"Age×Treatment interaction (β₃): {weights[3]:.3f}")

# Interpretation
print("\nInterpretation:")
print(f"- Baseline log-odds: {weights[0]:.3f}")
print(f"- Each SD increase in age changes log-odds by: {weights[1]:.3f}")
print(f"- Treatment changes log-odds by: {weights[2]:.3f}")
print(f"- Treatment effect varies with age by: {weights[3]:.3f} per SD")

# Predictions for specific scenarios
scenarios = [
    ("Young (30) + Placebo", -1.5, 0),
    ("Young (30) + Drug", -1.5, 1),
    ("Old (65) + Placebo", 1.5, 0),
    ("Old (65) + Drug", 1.5, 1)
]

print("\nPredictions:")
for desc, age_std, treatment in scenarios:
    x_new = np.array([1, age_std, treatment, age_std * treatment])
    log_odds = x_new @ weights
    prob = sigmoid(log_odds)
    print(f"{desc}: {prob:.3f}")
```

Example 5: Step-by-Step Manual Calculation

Problem: Weather and Sales

Does temperature affect ice cream sales?

Data:

```
Temp(°C): 15, 20, 25, 30
Sold out: No, No, Yes, Yes
Trials:   1,  1,  1,  1
```

Step 1: Standardize temperature

Mean = 22.5, SD = 6.45 x = [-1.16, -0.39, 0.39, 1.16]

Step 2: Calculate likelihood at α=-0.5, β=2

For each observation:

1. $x_1=-1.16$:
   - $f_1 = -0.5 + 2\times(-1.16) = -2.82$
   - $\theta_1 = \sigma(-2.82) \approx 0.056$
   - $L_1 = (1-0.056)^{1-0} = 0.944$
2. $x_2=-0.39$:
   - $f_2 = -0.5 + 2\times(-0.39) = -1.28$
   - $\theta_2 = \sigma(-1.28) \approx 0.218$
   - $L_2 = (1-0.218)^{1-0} = 0.782$
3. $x_3=0.39$:
   - $f_3 = -0.5 + 2\times(0.39) = 0.28$
   - $\theta_3 = \sigma(0.28) \approx 0.569$
   - $L_3 = (0.569)^1 = 0.569$
4. $x_4=1.16$:
   - $f_4 = -0.5 + 2\times(1.16) = 1.82$
   - $\theta_4 = \sigma(1.82) \approx 0.860$
   - $L_4 = (0.860)^1 = 0.860$

Total likelihood = $0.944 \times 0.782 \times 0.569 \times 0.860 = 0.361$

Step 3: Prediction for 27°C

- Standardized: $x^* = (27-22.5)/6.45 = 0.70$
- $f^* = -0.5 + 2\times0.70 = 0.90$
- $\theta^* = \sigma(0.90) \approx 0.711$
- Probability of selling out: 71.1%

Summary: Key Steps for Logistic Regression

1. **Prepare data**
   - Identify outcome (y) and covariates (x)
   - Standardize continuous covariates
   - Check sample sizes
2. **Specify model**
   - Choose link function (usually logit)
   - Include relevant covariates

- Consider interactions

3. **Estimate parameters**
   - MLE via optimization
   - Bayesian via MCMC/grid
   - Check convergence
4. **Make predictions**
   - Calculate θ(x) for new data
   - Compute credible/confidence intervals
   - Transform to probability scale
5. **Interpret results**
   - Coefficient signs and magnitudes
   - Odds ratios
   - Effect plots
6. **Validate model**
   - Check residuals
   - Cross-validation
   - Predictive performance