Logistic Regression Demystified

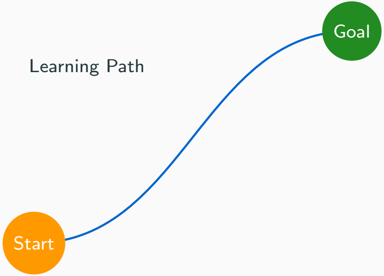
From Linear Models to Probabilistic Classification

Noel Jeffrey Pinton
CMSC 173 Machine Learning

Department of Computer Science
University of the Philippines - Cebu

Today's Journey

- **The Problem:** Why Classification?
- **The Solution:** Sigmoid Function
- **The Math:** How It Works
- **The Learning:** Training Process
- **The Assessment:** Model Evaluation



The Classification Challenge

What is Classification?

Predicting **discrete categories** rather than continuous values:

- Email: Spam or Not Spam?
- Medical: Benign or Malignant?
- Customer: Will Buy or Won't Buy?
- Image: Cat or Dog?

The Classification Challenge

What is Classification?

Predicting **discrete categories** rather than continuous values:

- Email: Spam or Not Spam?
- Medical: Benign or Malignant?
- Customer: Will Buy or Won't Buy?
- Image: Cat or Dog?

Why Linear Regression Fails

- Outputs any real number ($-\infty$ to $+\infty$)
- We need probabilities (0 to 1)
- Poor fit for binary outcomes
- Nonsensical predictions like 1.7 or -0.3

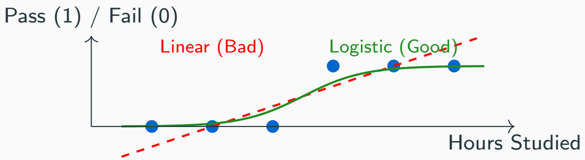


Figure 1: Linear vs. Logistic fit for binary classification

Linear vs. Logistic Regression: Side by Side

Linear Regression

Goal: Predict continuous values

Model:

$$y = \beta_0 + \beta_1 x_1 + \varepsilon$$

Output Range: $(-\infty, +\infty)$

Loss Function: Mean Squared Error

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Best for: Predicting house prices, temperature, sales revenue

Logistic Regression

Goal: Predict probabilities/categories

Model:

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1)}}$$

Output Range: $[0, 1]$

Loss Function: Cross-Entropy

$$CE = -\frac{1}{n}\sum_{i=1}^{n}[y_i \ln(p_i) + (1 - y_i)\ln(1 - p_i)]$$

Best for: Medical diagnosis, spam detection, customer churn

The key difference: Output constraints and loss functions

The Magic: Sigmoid Function

The Transformation

Step 1: Linear combination

$$z = \beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n$$

Step 2: Sigmoid transformation

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

Key Properties

- When $z = 0 \Rightarrow p = 0.5$
- When $z \to +\infty \Rightarrow p \to 1$
- When $z \to -\infty \Rightarrow p \to 0$
- S-shaped curve (monotonic)
- Smooth and differentiable

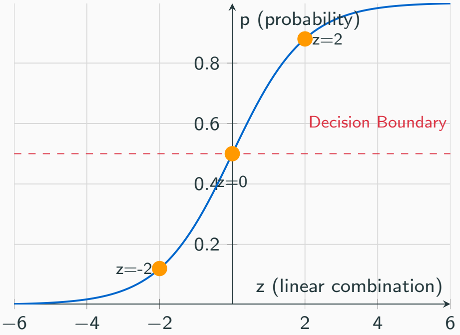


Figure 2: The Sigmoid Function: Mapping any real number to [0,1]

Interpreting Coefficients: The Odds Story

From Probabilities to Odds

The linear part gives us **log-odds**:

$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n$$

Coefficient Interpretation

Odds Ratio: $OR = e^{\beta_i}$

- If $OR > 1$: Positive association (increases odds)
- If $OR < 1$: Negative association (decreases odds)
- If $OR = 1$: No association

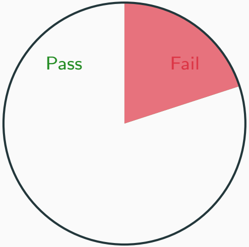
Real Example

Scenario: Predicting exam success

Model: $\beta_0 = -2.5, \beta_1 = 0.8$ (hours studied)

Interpretation: $e^{0.8} = 2.23$

Each additional study hour multiplies the odds of passing by 2.23!



Odds $= 0.69/0.31 = 2.23$

For 3 hours of study

Figure 3: Visual representation of odds

Training: The Cost Function

Cross-Entropy Loss

$$J(\beta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\ln(p^{(i)}) + (1 - y^{(i)})\ln(1 - p^{(i)})\right]$$

When $y = 1$ (True Positive)

Cost: $-\ln(p)$

- If $p \approx 1$: Cost ≈ 0
- If $p \approx 0$: Cost $\to \infty$

Message: "Be confident when you're right!"

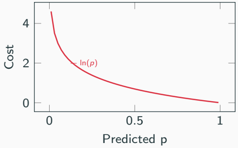


Figure 4: Cost when true class = 1

When $y = 0$ (True Negative)

Cost: $-\ln(1 - p)$

- If $p \approx 0$: Cost ≈ 0
- If $p \approx 1$: Cost $\to \infty$

Message: "Don't be confident when you're wrong!"

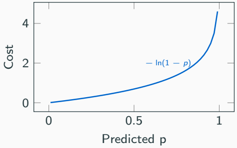


Figure 5: Cost when true class = 0

Worked Example: Step by Step

Dataset

Problem: Predict exam pass/fail based on study hours

Hours	0.5	1.5	2.5	3.5	4.5	5.5	6.5
Pass	0	0	0	1	1	1	1

Calculations

Initial guess: $\beta_0 = -2, \beta_1 = 0.6$

For student with 3.5 hours (y=1):

Step 1: $z = -2 + 0.6(3.5) = 0.1$

Step 2: $p = \frac{1}{1+e^{-0.1}} = 0.525$

Step 3: $\text{Cost} = -1 \cdot \ln(0.525) = 0.645$

Interpretation: 52.5

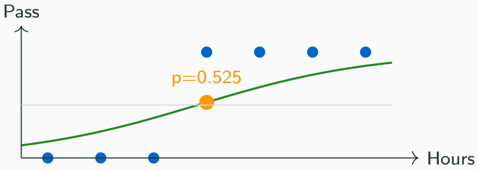


Figure 6: Model prediction for our example

Learning Algorithm: Gradient Descent

The Journey to Optimization

Goal: Find β that minimizes $J(\beta)$

Method: Follow the steepest descent

Update Rule:

$$\beta_j := \beta_j - \alpha \frac{\partial J}{\partial \beta_j}$$

where α is the learning rate

The Gradient

For logistic regression:

$$\frac{\partial J}{\partial \beta_j} = \frac{1}{m} \sum_{i=1}^{m} (p^{(i)} - y^{(i)}) x_j^{(i)}$$

Intuition: Adjust based on prediction errors!

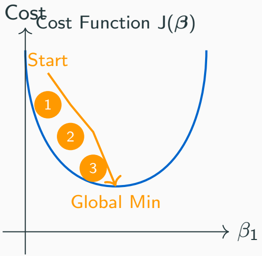


Figure 7: Gradient descent finds the minimum cost

Algorithm Steps

1. Initialize β randomly
2. Calculate predictions $p^{(i)}$
3. Compute cost $J(\beta)$
4. Update parameters using gradients
5. Repeat until convergence

Model Evaluation: Beyond Accuracy

Confusion Matrix

		Predicted	
		Pos	Neg
Actual	Pos	**TP**	FN
	Neg	FP	**TN**

Key Metrics

Accuracy: $\frac{TP+TN}{Total}$
Overall correctness

Precision: $\frac{TP}{TP+FP}$
"Of my positive predictions, how many were right?"

Recall: $\frac{TP}{TP+FN}$
"Of all actual positives, how many did I find?"

F1-Score: $\frac{2\times\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$

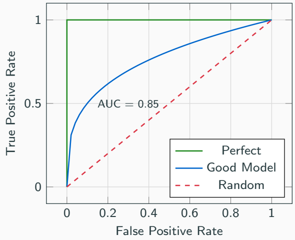


Figure 8: ROC Curve shows trade-off at all classification thresholds

Which Metric to Choose?

- **Medical diagnosis:** High Recall (catch all diseases)
- **Spam detection:** High Precision (avoid blocking good emails)
- **Balanced problems:** F1-Score or AUC

ROC Curve & AUC Explained

What is ROC Curve?

ROC = Receiver Operating Characteristic

Purpose: Shows classifier performance across all classification thresholds

Axes:

- **X-axis:** False Positive Rate = $\frac{FP}{FP+TN}$
- **Y-axis:** True Positive Rate = $\frac{TP}{TP+FN}$

Understanding AUC

AUC = Area Under the Curve

Interpretation:

- AUC = 1.0: Perfect classifier
- AUC = 0.5: Random guessing
- AUC < 0.5: Worse than random (invert predictions!)
- AUC > 0.8: Generally good model

Intuition: Probability that model ranks a random positive example higher than a random negative example

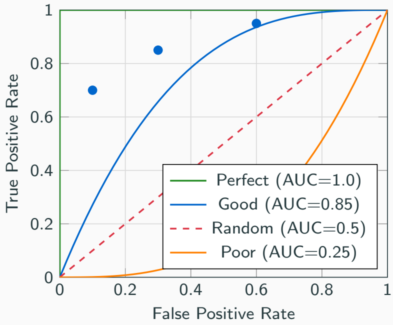


Figure 9: Different classifier performances on ROC space

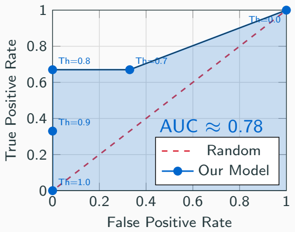
Why AUC is Useful

- **Threshold-independent:** Single number summary
- **Scale-invariant:** Measures prediction quality
- **Class-imbalance robust:** Works with unequal classes

Creating ROC Curve: Step-by-Step Process

Steps to Build ROC Curve

1. **Train your logistic regression model** and get predicted probabilities
2. **Sort predictions** by probability (highest to lowest)
3. **Initialize** TPR $= 0$, FPR $= 0$, start with threshold $= 1.0$
4. **For each unique threshold** (probability value):
   - Classify: if $p \geq \textit{threshold}$ then positive, else negative
   - Calculate confusion matrix (TP, FP, TN, FN)
   - Compute TPR $= \frac{TP}{TP+FN}$ and FPR $= \frac{FP}{FP+TN}$
   - Plot point (FPR, TPR)
5. **Connect the points** to form the ROC curve
6. **Calculate AUC** using trapezoidal rule



Example Data

Sample	True	Prob	Rank
A	1	0.95	1
B	1	0.85	2
C	0	0.75	3
D	1	0.65	4
E	0	0.45	5
F	0	0.25	6

Key Thresholds

- **Threshold 1.0:** All negative
  TPR=0, FPR=0
- **Threshold 0.9:** A positive
  TPR=1/3, FPR=0
- **Threshold 0.8:** A,B positive
  TPR=2/3, FPR=0
- **Threshold 0.7:** A,B,C positive
  TPR=2/3, FPR=1/3
- **Threshold 0.0:** All positive
  TPR=1, FPR=1

Performance Metrics: The Complete Picture

Primary Metrics

Accuracy: $\frac{TP+TN}{TP+TN+FP+FN}$

- Overall correctness
- Good for balanced datasets
- Can be misleading with imbalanced data

Precision (Positive Predictive Value): $\frac{TP}{TP+FP}$

- "When I predict positive, how often am I right?"
- Important when false positives are costly

Recall (Sensitivity/True Positive Rate): $\frac{TP}{TP+FN}$

- "Of all actual positives, how many did I catch?"
- Important when false negatives are costly

Composite Metrics

F1-Score: $\frac{2 \times Precision \times Recall}{Precision + Recall}$

- Harmonic mean of precision and recall
- Balances both false positives and negatives
- Good single metric for imbalanced data

Specificity (True Negative Rate): $\frac{TN}{TN+FP}$

- "Of all actual negatives, how many did I correctly identify?"
- Complement of False Positive Rate

Choosing the Right Metric

Medical Screening: Maximize Recall (don't miss diseases)
Spam Detection: Balance Precision and Recall (F1-Score)
Fraud Detection: Maximize Precision (minimize false alarms)
Balanced Data: Accuracy and AUC work well

Example 1: Student Exam Success Prediction

Dataset: Predicting Pass/Fail based on Study Hours

Student	A	B	C	D	E	F	G	H	I
Hours Studied	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
Actual Result	0	0	0	1	0	1	1	1	1
Predicted Prob	0.12	0.18	0.27	0.38	0.51	0.63	0.74	0.83	0.89

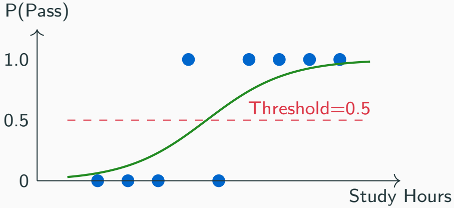


Figure 11: Logistic Regression Curve

Confusion Matrix (Threshold = 0.5)

		Predicted	
		Pass	Fail
Actual	Pass	**4**	1
	Fail	1	**3**

Predictions: Students E,F,G,H,I predicted to pass (prob ≥ 0.5)

Errors: Student E (FP), Student D (FN)

Performance Metrics

Accuracy: $\frac{4+3}{9} = 0.78$ (78%)

Precision: $\frac{4}{4+1} = 0.80$ (80%)

F1-Score: $\frac{2\times0.8\times0.8}{0.8+0.8} = 0.80$ (80%)

Recall: $\frac{4}{4+1} = 0.80$ (80%)

Example 2: Medical Diagnosis - Tumor Classification

Dataset: Predicting Malignant/Benign based on Tumor Size (cm)

Patient	1	2	3	4	5	6	7	8	9
Tumor Size	1.2	1.8	2.1	2.5	2.9	3.2	3.6	4.1	4.5
Actual (1=Malignant)	0	0	0	0	1	1	1	1	1
Predicted Prob	0.08	0.15	0.21	0.32	0.48	0.67	0.81	0.92	0.96

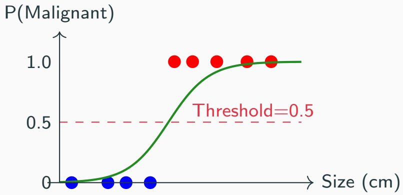


Figure 12: Tumor Classification Curve

Confusion Matrix (Threshold = 0.5)

		Predicted	
		Malignant	Benign
Actual	Malignant	4	1
	Benign	0	4

Predictions: Patients 6,7,8,9 predicted malignant (prob ≥ 0.5)
Error: Patient 5 missed (FN) - concerning in medical context!

Performance Metrics

Accuracy: $\frac{4+4}{9} = 0.89$ **Precision:** $\frac{4}{4+0} = 1.00$ **Recall:** $\frac{4}{4+1} = 0.80$ **F1-Score:** $\frac{2\times1.0\times0.8}{1.0+0.8} = 0.89$

Medical Context

High precision (no false alarms) but concerning recall (missed 20% of malignant cases). In medical diagnosis, we typically prioritize high recall to avoid missing diseases.

Example 3: Email Spam Detection

Dataset: Spam Classification based on Number of Suspicious Words

Email	1	2	3	4	5	6	7	8	9	10
Suspicious Words	0	1	1	2	3	3	4	5	6	7
Actual (1=Spam)	0	0	0	0	0	1	1	1	1	1
Predicted Prob	0.05	0.12	0.12	0.27	0.47	0.47	0.73	0.88	0.95	0.98

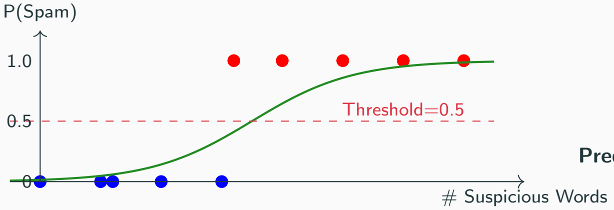


Figure 13: Spam Detection Curve

Confusion Matrix (Threshold = 0.5)

		Predicted	
		Spam	Ham
Actual	Spam	4	1
	Ham	1	4

Predictions: Emails 7,8,9,10 + Email 5 predicted spam (prob ≥ 0.5)

Errors: Email 5 (FP), Email 6 (FN)

Performance Metrics

Accuracy: $\frac{4+4}{10} = 0.80$ **Precision:** $\frac{4}{4+1} = 0.80$ **Recall:** $\frac{4}{4+1} = 0.80$ **F1-Score:** $\frac{2\times 0.8\times 0.8}{0.8+0.8} = 0.80$ (

Email Context

Balanced performance. False positive (legitimate email marked spam) can be as problematic as false negative (spam in inbox). F1-Score provides good overall assessment.

Threshold Selection: Making the Right Trade-offs

The Threshold Dilemma

Default threshold = 0.5, but this may not always be optimal!

Lower Threshold (e.g., 0.3):

- More positive predictions
- Higher Recall (catch more positives)
- Lower Precision (more false alarms)
- Good when missing positives is costly

Higher Threshold (e.g., 0.7):

- Fewer positive predictions
- Higher Precision (fewer false alarms)
- Lower Recall (miss more positives)
- Good when false alarms are costly

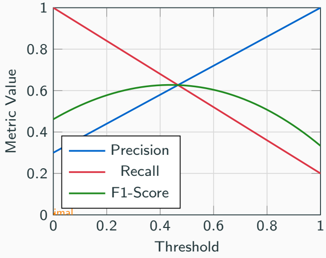


Figure 14: Precision-Recall trade-off vs. threshold

Domain-Specific Threshold Selection

- **Cancer Screening:** Use low threshold (0.2-0.3) to maximize recall
- **Fraud Detection:** Use high threshold (0.7-0.8) to minimize false alarms
- **Marketing Campaigns:** Use moderate threshold (0.4-0.6) for balanced approach
- **A/B Testing:** Optimize threshold based on business metrics

Practical Considerations & Tips

Common Pitfalls

- **Perfect Separation:** When classes are perfectly separable, coefficients explode
- **Multicollinearity:** Highly correlated features cause instability
- **Sample Size:** Need adequate samples per feature
- **Outliers:** Can strongly influence the model

Best Practices

- **Feature Scaling:** Standardize continuous variables
- **Regularization:** Use L1/L2 to prevent overfitting
- **Cross-validation:** Always validate on unseen data
- **Feature Selection:** Remove irrelevant variables

Quick Implementation

Python (sklearn):

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

# Create and train model
model = LogisticRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)
probabilities = model.predict_proba(X_test)

# Evaluate
print(classification_report(y_test, y_pred))
```

Extensions

- **Multinomial:** Multiple classes (2)
- **Ordinal:** Ordered categories
- **Regularized:** Ridge/Lasso logistic regression

Key Takeaways

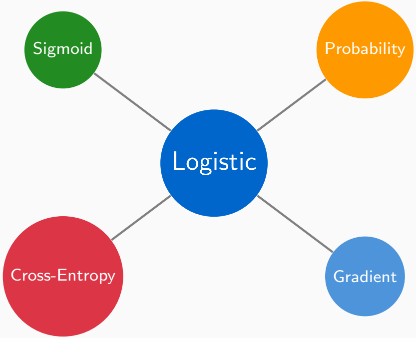
What We've Learned

1. **Purpose:** Binary classification with probabilistic output
2. **Mechanism:** Sigmoid function maps linear combinations to [0,1]
3. **Training:** Minimize cross-entropy loss using gradient descent
4. **Interpretation:** Coefficients represent log-odds ratios
5. **Evaluation:** Multiple metrics beyond accuracy
6. **ROC/AUC:** Threshold-independent performance assessment
7. **Trade-offs:** Precision vs. Recall based on domain needs

The Big Picture

Logistic regression is the **foundation** of many machine learning techniques:

- Neural networks use sigmoid activations
- Maximum likelihood estimation principles
- Probabilistic thinking in ML



From linear thinking to probabilistic reasoning

Questions?

Contact

nppinton@up.edu.ph

/in/njpinton

Further Reading

Books:

- Pattern Recognition and Machine Learning by Christopher Bishop

Online:

- Coursera ML Course
- Kaggle Learn

Thank you for your attention!