CSE 403: Machine Learning

Chapter 4: Logistic Regression

Md Atikuzzaman
Lecturer
Department of Computer Science & Engineering
atik@cse.green.edu.bd

GREEN UNIVERSITY OF BANGLADESH

Outline

1. Logistic Regression: Motivation
2. Model: Linear Score to Probability
3. Log-Odds Interpretation
4. Gradients: Sigmoid Derivative
5. Decision Boundary Intuition
6. Loss Function and Objective
7. Training: Gradient Descent
8. Evaluation and Interpretation
9. Softmax Regression (Multi-Class)
10. Summary
11. References

Why Logistic Regression?

- Many real problems require predicting a **class label** (0/1), not a continuous value.
- Examples: spam detection, disease diagnosis, fraud detection, churn prediction.
- Logistic regression is a strong **baseline**: simple, fast, and interpretable.

Unlike linear regression, logistic regression outputs a **probability** and then applies a **decision threshold**.

Supervised Learning View (Binary Classification)

Training data

$$\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{m}, \qquad y^{(i)} \in \{0, 1\}$$

Goal

Learn a function $f_\theta(x)$ that outputs a probability:

$$\hat{p} = P(y = 1 \mid x)$$

We want calibrated probabilities when possible, not only hard labels.

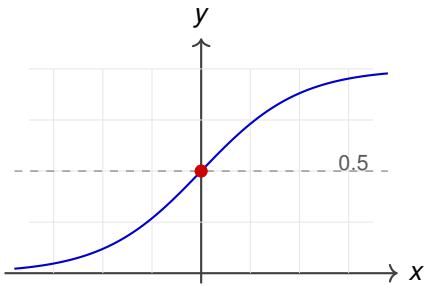
From Linear Score to Probability

Linear score

$$z = \theta_0 + \theta^\top x$$

Sigmoid (logistic) function

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



- $\sigma(z) \in (0, 1)$, so it can be interpreted as a probability.
- Large $z \Rightarrow$ probability near 1; small $z \Rightarrow$ near 0.

Odds and Log-Odds (Logit)

Odds of class 1

If $\hat{p} = P(y = 1 \mid x)$ then

$$\text{Odds} = \frac{\hat{p}}{1 - \hat{p}}$$

Log-odds (logit)

$$\log\left(\frac{\hat{p}}{1 - \hat{p}}\right)$$

- Converts probability $(0, 1)$ into a real value $(-\infty, \infty)$.
- Lets us keep a **linear model** in the transformed space.

Interpretation: logistic regression is **linear in log-odds**, not linear in probability.

Logistic Regression as a Log-Odds Model

Model assumption

$$\log\left(\frac{P(y=1 \mid x)}{1-P(y=1 \mid x)}\right) = \theta_0 + \theta^\top x$$

- Each coefficient θ_j adds to the **log-odds** of class 1.
- This is why logistic regression is considered interpretable.

A 1-unit increase in x_j changes the log-odds by θ_j (holding other features fixed).

From Log-Odds to the Sigmoid Function

Logistic regression models the **log-odds** as

$$\begin{aligned}
\log\left(\frac{p}{1-p}\right) &= z \\
\frac{p}{1-p} &= e^z \\
p &= e^z(1-p) \\
p + e^z p &= e^z \\
p(1+e^z) &= e^z \\
p &= \frac{e^z}{1+e^z} \\
p &= \frac{1}{1+e^{-z}}
\end{aligned}$$

The sigmoid function naturally arises from modeling the **log-odds** as a linear function.

Derivative of the Sigmoid

Sigmoid

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

Derivative (useful identity)

$$\frac{d\sigma(z)}{dz} = \sigma(z)\big(1 - \sigma(z)\big)$$

- Depends only on the sigmoid value itself.
- Makes gradient-based optimization efficient.

Logistic Regression Hypothesis and Prediction

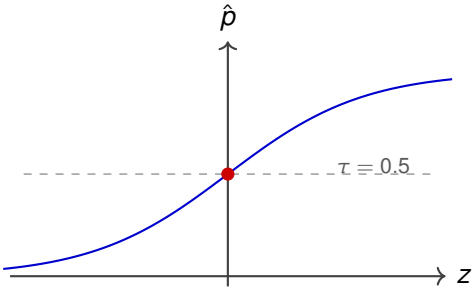
Probability model

$$\hat{p} = f_\theta(x) = \sigma(\theta_0 + \theta^\top x)$$

Prediction rule (threshold)

$$\hat{y} = \begin{cases} 1, & \hat{p} \geq \tau \\ 0, & \hat{p} < \tau \end{cases}$$

Choose τ based on cost: in fraud/medical tasks, $\tau = 0.5$ is not always best.



Decision Boundary

Key idea

For $\tau = 0.5$:

$$\hat{p} = 0.5 \;\Leftrightarrow\; \sigma(z) = 0.5 \;\Leftrightarrow\; z = 0$$

Boundary equation

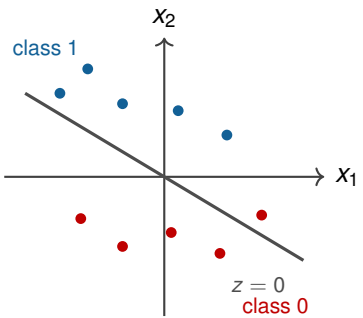
$$\theta_0 + \theta^\top x = 0$$

- Produces a **linear** decision boundary in the original feature space.
- With feature mapping, it can represent nonlinear boundaries.

Geometric View (2D Example)

- In 2D: $x = (x_1, x_2)$.
- The line $\theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0$ splits the plane.
- One side predicts class 1 (higher $\hat{p}$), the other class 0.

The sigmoid changes confidence smoothly, but the boundary at $z = 0$ is a straight line.



Why Not Use MSE for Classification?

- MSE is natural for regression, but for classification it often gives:
  - weaker probability learning,
  - slower or unstable optimization.
- Logistic regression uses **cross-entropy (log loss)** designed for probability models.

Cross-entropy strongly penalizes confident wrong predictions, which is exactly what we want.

Binary Cross-Entropy (Log Loss)

Loss for one example

For $y \in \{0, 1\}$ and $\hat{p} \in (0, 1)$:

$$\ell(\hat{p}, y) = -\Big(y \log(\hat{p}) + (1 - y) \log(1 - \hat{p})\Big)$$

- If $y = 1$: $\ell = -\log(\hat{p})$ (small if $\hat{p}$ is near 1).
- If $y = 0$: $\ell = -\log(1 - \hat{p})$ (small if $\hat{p}$ is near 0).

Empirical Risk (Training Objective)

Objective

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \ell\Big(f_\theta(x^{(i)}), y^{(i)}\Big)$$

Expanded form

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \Big(y^{(i)} \log(\hat{p}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{p}^{(i)})\Big)$$

Training = choose θ that makes true labels likely under the model.

Gradient Descent for Logistic Regression

Update rule

$$\theta \leftarrow \theta - \alpha \nabla_\theta J(\theta)$$

Gradient (vector form)

Let $\hat{p} = \sigma(X\theta)$:

$$\nabla_\theta J(\theta) = \frac{1}{m} X^\top (\hat{p} - y)$$

- Similar structure to linear regression, but uses probabilities.
- Works with batch GD, SGD, and mini-batch GD.

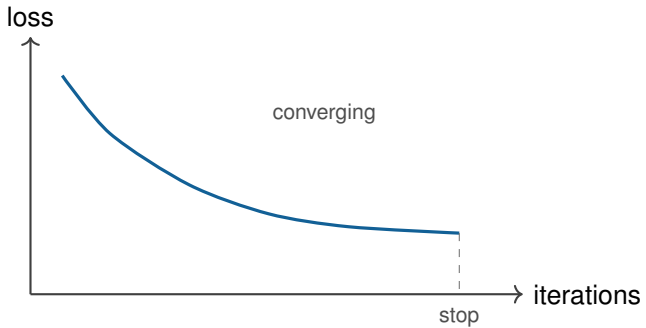
Practical Training Tips

- **Feature scaling** often improves convergence speed.
- **Class imbalance:** tune threshold, use class weights, or resampling.
- **Regularization** reduces overfitting (common when features are many):

$$J_\lambda(\theta) = J(\theta) + \lambda \|\theta\|_2^2$$

Regularization improves generalization, especially with correlated features.

Typical Training Curve (Illustration)



Stop using a max-iteration rule or when validation loss stops improving (early stopping).

How Do We Evaluate a Classifier?

- **Accuracy:** $\frac{TP+TN}{TP+TN+FP+FN}$
- **Precision:** $\frac{TP}{TP+FP}$
- **Recall:** $\frac{TP}{TP+FN}$
- **F1-score:** harmonic mean of precision and recall

For imbalanced data, accuracy can be misleading. Precision/recall and F1 are usually better.

Confusion Matrix (Binary Classification)

	Pred 1	**Pred 0**
True 1	*TP*	*FN*
True 0	*FP*	*TN*

- Helps you see **which** mistakes are happening.
- Most classification metrics come directly from this table.

Interpreting Coefficients with Odds

Log-odds form

$$\log\left(\frac{P(y=1 \mid x)}{1-P(y=1 \mid x)}\right) = \theta_0 + \theta^\top x$$

- If $\theta_j > 0$, increasing x_j increases the probability of class 1.
- If $\theta_j < 0$, increasing x_j decreases the probability of class 1.
- Exponentiating gives an **odds ratio**: e^{θ_j} (per 1-unit increase in x_j).

Interpretation is strongest when features are well-defined and properly scaled.

Interpreting the Prediction

- Predicted probability

$$\hat{p} = 0.5$$

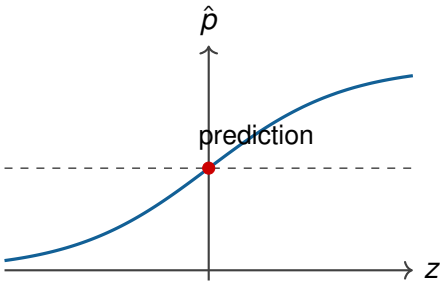
- If decision threshold $\tau = 0.5$

$$\hat{y} = \begin{cases} 1 & \text{if } \hat{p} \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

- Therefore

$$\hat{y} = 1$$

The student is predicted to **pass the exam**.



Why Do We Need Softmax?

- Logistic regression works for **binary** classification: $y \in \{0, 1\}$.
- Many real tasks are **multi-class**:

$$y \in \{1, 2, \ldots, K\}$$

- Example: digit recognition (0–9), topic classification, disease type prediction.

Softmax is the standard way to convert K raw scores into a valid probability distribution that sums to 1.

Softmax Model (From Scores to Probabilities)

Linear scores (logits)

For K classes, compute a score for each class:

$$z_k = \theta_k^\top x \quad \text{for } k = 1, \ldots, K$$

Let $z = [z_1, \ldots, z_K]^\top$.

Softmax probabilities

$$p_k = P(y = k \mid x) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$$

- $p_k \in (0, 1)$ for each class.

Softmax: Tiny Numerical Example

Assume $K = 3$ classes and the model produces logits:

$$z = [2,\ 1,\ 0]$$

Compute exponentials and normalize

$$e^z = [e^2,\ e^1,\ e^0] \approx [7.39,\ 2.72,\ 1]$$

$$\sum_{j=1}^{3} e^{z_j} \approx 7.39 + 2.72 + 1 = 11.11$$

$$p = \mathrm{softmax}(z) \approx \left[\frac{7.39}{11.11},\ \frac{2.72}{11.11},\ \frac{1}{11.11}\right] \approx [0.665,\ 0.245,\ 0.090]$$

Prediction = class 1 (highest probability).

Categorical Cross-Entropy Loss

One-hot target

For K classes, represent the true label as a one-hot vector:

$$y \in \{0,1\}^K, \quad \sum_{k=1}^{K} y_k = 1$$

Categorical cross-entropy (single example)

If $p_k = P(y = k \mid x)$, then

$$\ell(p, y) = -\sum_{k=1}^{K} y_k \log(p_k)$$

Categorical Cross-Entropy Loss

Example

If true class is 1, then $y = [1, 0, 0]$ and

$$\ell = -\log(p_1)$$

Using the previous example $p_1 \approx 0.665$:

$$\ell \approx -\log(0.665) \approx 0.41$$

Cross-entropy focuses learning on increasing the probability of the true class.

Why Softmax + Cross-Entropy Works So Well

- Softmax gives a **proper probability distribution** over K classes.
- Cross-entropy is the right objective when we want the model to assign **high probability to the true class**.
- Strong penalty for confident wrong predictions:

$$\text{if } p_{\text{true}} \to 0, \quad -\log(p_{\text{true}}) \to \infty$$

- Training is efficient with gradient-based methods (GD/SGD/mini-batch).

Softmax turns scores into probabilities; cross-entropy teaches the model to make the true class most probable.

Linear vs Logistic vs Softmax Regression

Aspect	Linear	Logistic (Binary)	Softmax (Multi-class)
Output	Continuous	Probability (0–1)	Probabilities (sum to 1)
Target y	$y \in \mathbb{R}$	$y \in \{0, 1\}$	$y \in \{1, \ldots, K\}$
Model	$y = \theta^\top x$	$\sigma(\theta^\top x)$	$p_k = \frac{e^{z_k}}{\sum_j e^{z_j}}$
Loss	MSE	Binary CE	Categorical CE
Decision	—	threshold τ	$\arg\max_k p_k$
Metrics	RMSE, MAE, R^2	Acc, Prec, Rec, F1	Acc, Macro-F1, Conf. matrix

Softmax regression is the standard baseline for **multi-class** classification.

Key Takeaways

- **Logistic regression** is a strong baseline for **binary classification** and outputs a probability: $\hat{p} = \sigma(\theta_0 + \theta^\top x)$
- Logistic regression is **linear in log-odds**: $\log\left(\frac{p}{1-p}\right) = \theta_0 + \theta^\top x$
- Training uses **cross-entropy (log loss)** which strongly penalizes confident wrong predictions.
- The **decision boundary** (for $\tau = 0.5$) is where $z = 0$: $\theta_0 + \theta^\top x = 0$
- For **multi-class problems**, use **softmax regression** to produce a valid probability distribution: $p_k = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$
- Multi-class training uses **categorical cross-entropy**: $\ell(p, y) = -\sum_{k=1}^{K} y_k \log(p_k)$
- Evaluate classification models using **accuracy, precision, recall, F1**, and prefer precision/recall for imbalanced datasets.

Binary: Sigmoid + Binary Cross-Entropy **Multi-class: Softmax + Categorical Cross-Entropy**

References

1. T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning*, Springer.
2. C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer.
3. I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, MIT Press.