Artificial Intelligence

Lecture 9: Logistic Regression — Deep Dive & Evaluation

Prof. Antonio Khalil Moretti

Week 10

SCIS 432
Spelman College

Today's Agenda

Part 1 — Foundations Review

1. Logistic regression recap
2. Binomial distribution & likelihood
3. Cross-entropy: derivation from MLE
4. Log-odds & odds ratio — worked examples
5. Check your understanding

Part 2 — Gradient Derivation

6. Chain rule derivation of the gradient
7. Vectorized gradient update

Part 3 — Evaluation

8. Beyond accuracy: confusion matrix
9. Precision, recall, F1-score
10. Imbalanced datasets
11. Check your understanding

Part 4 — Extensions

12. Multi-class logistic regression (softmax)
13. Summary & next steps

Logistic Regression: Quick Recap

Everything in one place

The model — three steps

Step 1: Linear score

$$z = \mathbf{w}^T\mathbf{x} \in \mathbb{R} \quad \text{(bias absorbed)}$$

Step 2: Sigmoid squashing

$$\hat{p} = \sigma(z) = \frac{1}{1+e^{-z}} \in (0,1)$$

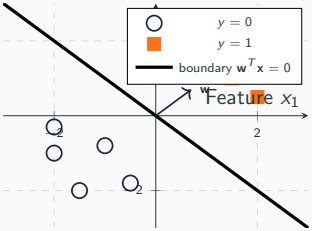
Step 3: Hard prediction

$$\hat{y} = \mathbf{1}[\hat{p} \geq 0.5] = \mathbf{1}[z \geq 0]$$

Key identity — log-odds:

$$\log\left(\frac{\hat{p}}{1-\hat{p}}\right) = z = \mathbf{w}^T\mathbf{x}$$

Geometric picture



Decision boundary is a **linear hyperplane**. $\mathbf{w}$ is normal to it.

The Binomial Distribution

The probability model behind logistic regression

Definition

A **Bernoulli trial** is a single experiment with two outcomes: **success** ($y = 1$) with probability p, and **failure** ($y = 0$) with probability $1 - p$.

The **Bernoulli distribution** gives:

$$P(y \mid p) = p^y (1-p)^{1-y}, \quad y \in \{0, 1\}$$

The **Binomial distribution** counts k successes in n independent Bernoulli trials:

$$P(k \mid n, p) = \binom{n}{k} p^k (1-p)^{n-k}$$

Mean: $\mu = np$

Variance: $\sigma^2 = np(1-p)$

Coin flip example

Suppose we flip a (possibly biased) coin $n = 5$ times.

$p = P(\text{heads})$, $y_i = 1$ if heads, 0 if tails.

Observed sequence: H, T, H, H, T ($k = 3$ heads)

Likelihood of this exact sequence:

$$P(y_1, \ldots, y_5 \mid p) = p \cdot (1-p) \cdot p \cdot p \cdot (1-p)$$

$$= p^3 (1-p)^2$$

Likelihood of exactly 3 heads (any order):

$$P(k{=}3 \mid n{=}5, p) = \binom{5}{3} p^3 (1-p)^2 = 10\, p^3 (1-p)^2$$

Key question: What value of p makes this data

MLE for the Coin: A Concrete Calculation

Finding the most likely probability from data

Likelihood: $\mathcal{L}(p) = p^3(1-p)^2$

Log-likelihood (easier to differentiate):

$$\ell(p) = 3\log p + 2\log(1-p)$$

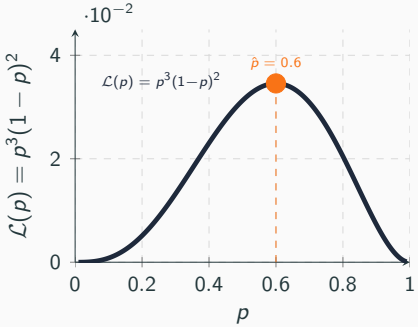
Set derivative to zero:

$$\frac{d\ell}{dp} = \frac{3}{p} - \frac{2}{1-p} = 0$$

$$3(1-p) = 2p \;\Rightarrow\; 3 - 3p = 2p \;\Rightarrow\; \hat{p}_{\text{MLE}} = \frac{3}{5} = 0.6$$

General result: For k heads in n flips,

$$\hat{p}_{\text{MLE}} = \frac{k}{n} = \text{sample proportion}$$



The likelihood is maximized exactly at the sample proportion — as expected.

From Binomial Likelihood to Cross-Entropy Loss

Why cross-entropy is the right loss for classification

Step-by-step derivation

Each label $y_i \in \{0, 1\}$ follows a Bernoulli with $\hat{p}_i = \sigma(\mathbf{w}^T \mathbf{x}_i)$.

1. **Likelihood of one point:**

$$P(y_i \mid \mathbf{x}_i, \mathbf{w}) = \hat{p}_i^{y_i}(1 - \hat{p}_i)^{1-y_i}$$

2. **Joint likelihood (independence):**

$$\mathcal{L}(\mathbf{w}) = \prod_{i=1}^{n} \hat{p}_i^{y_i}(1 - \hat{p}_i)^{1-y_i}$$

3. **Log-likelihood:**

$$\ell(\mathbf{w}) = \sum_{i=1}^{n} \big[y_i \log \hat{p}_i + (1 - y_i) \log(1 - \hat{p}_i)\big]$$

4. **Negate & normalize to get a loss:**

$$J(\mathbf{w}) = -\frac{1}{n} \sum_{i=1}^{n} \big[y_i \log \hat{p}_i + (1 - y_i) \log(1 - \hat{p}_i)\big]$$

Connection to information theory

The cross-entropy between true distribution $\mathbf{y}$ and predicted $\hat{\mathbf{p}}$ is:

$$H(\mathbf{y}, \hat{\mathbf{p}}) = -\sum_i \big[y_i \log \hat{p}_i + (1 - y_i) \log(1 - \hat{p}_i)\big]$$

Three equivalent views of the loss:

Statistics:	Negative log-likelihood of Bernoulli model
Info theory:	Cross-entropy between true labels and predictions
Optimization:	Convex function of $\mathbf{w}$ $\Rightarrow$ unique global minimum

Maximize $\ell(\mathbf{w})$ $\equiv$ Minimize $J(\mathbf{w})$
(same solution, opposite sign)

Interpreting Logistic Regression Coefficients

Log-odds, odds ratios, and what the weights actually mean

The log-odds identity

Recall: $\hat{p} = \sigma(\mathbf{w}^T\mathbf{x})$, so:

$$\underbrace{\log\left(\frac{\hat{p}}{1-\hat{p}}\right)}_{\text{log-odds (logit)}} = \mathbf{w}^T\mathbf{x} = w_1x_1 + \cdots + w_dx_d$$

What each weight means:
Holding all other features fixed, increasing x_j by **1 unit**:

- **adds** w_j to the log-odds
- **multiplies** the odds by e^{w_j}

The odds ratio for a 1-unit change in x_j:

$$\text{OR}_j = e^{w_j}$$

Worked example — credit default model

Suppose a fitted model has:

$$z = -3.5 + 0.04\,x_1 - 0.8\,x_2 + 1.6\,x_3$$

where x_1 = age (years), x_2 = credit score (hundreds), x_3 = missed payments.

Feature	w_j	e^{w_j}	Interpretation
Age (x_1)	0.04	1.04	Each year: +4% odds
Credit (x_2)	-0.8	0.45	+100 pts: odds $\times 0.45$
Missed (x_3)	1.6	4.95	Each miss: $\approx 5\times$ odds
Intercept	-3.5	—	Baseline log-odds

Prediction example:

Check Your Understanding: Interpreting Coefficients

Question 1 — Odds & Probabilities

A logistic model outputs $\hat{p} = 0.75$.

(a) What are the **odds** of class 1?

(b) What is the **log-odds**?

(c) A second patient has log-odds $= -2$. Compute $\hat{p}$ for this patient.

Hint: $\sigma(z) = 1/(1 + e^{-z})$

Question 2 — Odds Ratios

A tumor-detection model has $w_{\text{size}} = 0.3$.

(a) What is the odds ratio for a 1 cm increase in tumor size?

(b) A tumor grows from 2 cm to 5 cm. By

Question 3 — Model Interpretation

A spam classifier has:

$$z = -2 + 1.5\,x_1 + 0.2\,x_2 - 3.1\,x_3$$

where x_1 = number of exclamation marks, x_2 = email length (hundreds of words), $x_3 = 1$ if from known sender.

(a) Which single feature has the largest impact on the odds ratio?

(b) Compute e^{w_3}. What does this mean in plain English?

(c) An email has 3 exclamation marks, length 2 (hundred words), from unknown sender. Compute z and $\hat{p}$, then classify.

Deriving the Gradient: Setup

Applying the chain rule to the cross-entropy loss

Goal

We want to compute $\frac{\partial J}{\partial \mathbf{w}}$ where:

$$J(\mathbf{w}) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log \hat{p}_i + (1-y_i)\log(1-\hat{p}_i)\right]$$

and $\hat{p}_i = \sigma(z_i)$, $z_i = \mathbf{w}^T\mathbf{x}_i$.

Computation graph for one point i:

By the **chain rule**:

$$\frac{\partial \ell_i}{\partial \mathbf{w}} = \underbrace{\frac{\partial \ell_i}{\partial \hat{p}_i}}_{\text{Term A}} \cdot \underbrace{\frac{d\hat{p}_i}{dz_i}}_{\text{Term B}} \cdot \underbrace{\frac{\partial z_i}{\partial \mathbf{w}}}_{\text{Term C}}$$

Each term

Term A — derivative of log-loss w.r.t. $\hat{p}_i$:

$$\frac{\partial \ell_i}{\partial \hat{p}_i} = -\frac{y_i}{\hat{p}_i} + \frac{1-y_i}{1-\hat{p}_i} = \frac{\hat{p}_i - y_i}{\hat{p}_i(1-\hat{p}_i)}$$

Term B — sigmoid derivative (proved last lecture):

$$\frac{d\hat{p}_i}{dz_i} = \sigma'(z_i) = \hat{p}_i(1-\hat{p}_i)$$

Term C — derivative of linear score:

$$\frac{\partial z_i}{\partial \mathbf{w}} = \mathbf{x}_i$$

Multiply Terms A $\times$ B:

$$\frac{\hat{p}_i - y_i}{\hat{p}_i(1-\hat{p}_i)} \cdot \hat{p}_i(1-\hat{p}_i) = \hat{p}_i - y_i$$

Deriving the Gradient: Putting It Together

From a single point to the full vectorized update

Gradient for one point i

Combining Terms A $\times$ B $\times$ C:

$$\frac{\partial \ell_i}{\partial \mathbf{w}} = (\hat{p}_i - y_i)\,\mathbf{x}_i$$

Gradient of the full loss (sum and divide by n):

$$\nabla_{\mathbf{w}} J = \frac{1}{n}\sum_{i=1}^{n}(\hat{p}_i - y_i)\,\mathbf{x}_i$$

Vectorized form:
Let $\hat{\mathbf{p}} = \sigma(\mathbf{X}\mathbf{w}) \in \mathbb{R}^n$. Then:

$$\nabla_{\mathbf{w}} J = \frac{1}{n}\mathbf{X}^T(\hat{\mathbf{p}} - \mathbf{y})$$

Gradient descent update:

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \frac{1}{n}\mathbf{X}^T(\hat{\mathbf{p}} - \mathbf{y})$$

Intuition

The gradient has a beautiful interpretation:

$$\underbrace{\mathbf{X}^T}_{\text{features}}\ \underbrace{(\hat{\mathbf{p}} - \mathbf{y})}_{\text{residuals}}$$

Each feature $\mathbf{x}_i$ is **weighted by the residual** $(\hat{p}_i - y_i)$:

- If $\hat{p}_i > y_i$ (over-predicted), we *decrease* $\mathbf{w}$ in the direction of $\mathbf{x}_i$
- If $\hat{p}_i < y_i$ (under-predicted), we *increase* $\mathbf{w}$
- If $\hat{p}_i = y_i$ (perfect), no update

Same form as linear regression!
Replace $\hat{\mathbf{p}} = \sigma(\mathbf{X}\mathbf{w})$ with $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$ and you get the linear regression gradient exactly.

Beyond Accuracy: The Confusion Matrix

Accuracy is often a misleading metric

Why accuracy can mislead

Consider a disease screening test where only 1% of people have the disease.

A model that *always predicts negative* achieves **99% accuracy** — yet is completely useless clinically.

The Confusion Matrix

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

- **TP** True Positive: correctly predicted positive
- **TN** True Negative: correctly predicted negative
- **FP** False Positive: predicted 1, actually 0 (Type I)

A concrete example

100 patients; 10 have disease. Model predicts:

	Pred: sick	**Pred: healthy**
True: sick	7 (TP)	3 (FN)
True: healthy	12 (FP)	78 (TN)

$$\text{Accuracy} = \frac{TP + TN}{n} = \frac{7 + 78}{100} = 85\%$$

But wait — we *missed* 3 sick patients (FN) and *alarmed* 12 healthy ones (FP)!

Different errors have different costs:

- In medicine: FN is catastrophic (missed diagnosis)
- In spam: FP is bad (losing real email)

Precision, Recall, and F1-Score

Definitions

Precision — of all predicted positives, how many are correct?

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall (Sensitivity) — of all actual positives, how many did we catch?

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1-Score — harmonic mean of precision and recall:

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Why harmonic mean? It punishes extreme imbalance

Computed on our example

Recall: TP=7, FP=12, FN=3, TN=78.

$$\text{Precision} = \frac{7}{7 + 12} = \frac{7}{19} \approx 0.368$$

$$\text{Recall} = \frac{7}{7 + 3} = \frac{7}{10} = 0.70$$

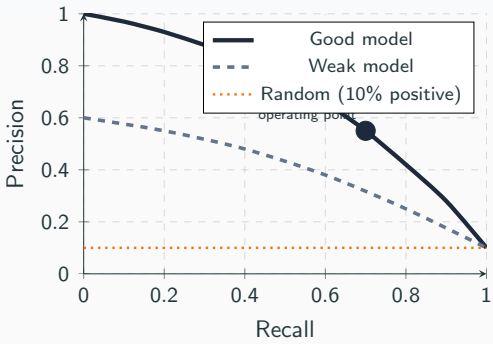
$$F_1 = \frac{2 \times 0.368 \times 0.70}{0.368 + 0.70} \approx 0.48$$

Precision–Recall tradeoff

Changing the decision threshold shifts the balance:

- **Lower threshold** ⇒ more positives predicted ⇒ higher recall, lower precision
- **Higher threshold** ⇒ fewer positives predicted ⇒ higher precision, lower recall

Visualizing the Precision-Recall Tradeoff



The PR curve

Each point on the curve corresponds to a different **decision threshold**.

Area Under the PR Curve (AUPRC):
A single number summarizing performance across all thresholds. Higher is better.

When to use PR vs. ROC curves:

- **PR curves:** preferred when the *positive class is rare* (imbalanced datasets)
- **ROC curves:** preferred when both classes matter equally

A perfect classifier would have AUPRC $= 1.0$ (top-right corner).
A random classifier achieves AUPRC $\approx$ fraction of positive examples.

Imbalanced Datasets

When one class vastly outnumbers the other

The problem

Many real datasets are highly imbalanced:

- Fraud detection: $\sim$0.1% fraud
- Disease screening: $\sim$1–5% positive
- Defect detection: $\sim$0.01% defective parts

A *naive* model predicting the majority class always achieves high accuracy but zero recall on the minority class.

Strategies to handle imbalance

Resampling:	Oversample minority (SMOTE) or undersample majority
Class weights:	Penalize misclassifying minority class more heavily

Class-weighted loss

Modify the cross-entropy to weight each class:

$$J = -\frac{1}{n}\sum_{i=1}^{n}\left[c_1\, y_i \log \hat{p}_i + c_0(1-y_i)\log(1-\hat{p}_i)\right]$$

Setting $c_1 = \dfrac{n}{2n_1}$ and $c_0 = \dfrac{n}{2n_0}$ balances the contribution of each class.

Check your understanding

1. A model trained on 1000 negatives and 10 positives achieves accuracy 99%. Is this a good model? What other metrics would you compute?
2. You lower the threshold from 0.5 to 0.2. Does precision go up or down? Does recall go up or down?

Check Your Understanding: Precision, Recall & F1

Question 1 — Confusion Matrix

A model is tested on 200 emails (50 spam, 150 not spam). Results:

$$\text{TP} = 40,\ \text{FP} = 20,\ \text{FN} = 10,\ \text{TN} = 130$$

(a) Compute accuracy, precision, recall, and F_1.

(b) The model flags a legitimate email as spam (FP). In what situation is a FP more costly than a FN?

(c) If you increase the threshold to 0.8, which of TP/FP/FN/TN will increase, and which will decrease?

Question 2 — F1 vs. Accuracy

A fraud dataset has 990 legitimate and 10

Question 3 — Threshold Analysis

Consider three operating points of the same model:

Threshold	Precision	Recall	F_1
0.3	0.40	0.90	?
0.5	0.65	0.70	?
0.8	0.90	0.30	?

(a) Compute F_1 for each threshold.

(b) Which threshold would you choose for a cancer screening test? For an email spam filter?

(c) Why is the harmonic mean used instead of the arithmetic mean for F_1?

Multi-Class Classification: Softmax Regression

Extending logistic regression to $K > 2$ classes

The softmax function

For K classes, compute one linear score per class:

$$z_k = \mathbf{w}_k^T \mathbf{x}, \quad k = 1, \ldots, K$$

Convert all K scores to probabilities:

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

The softmax is the multi-class generalization of sigmoid:

- Each $P(y = k) \in (0, 1)$
- All K probabilities sum to 1
- Predict the class with the highest probability

Connection to logistic regression:

Example: 3-class digit recognition

Suppose scores for an image are:

$$z_0 = 1.2 \quad \text{(digit 0)}$$

$$z_1 = 3.5 \quad \text{(digit 1)}$$

$$z_2 = 0.8 \quad \text{(digit 2)}$$

Denominator:
$e^{1.2} + e^{3.5} + e^{0.8} = 3.32 + 33.1 + 2.23 = 38.65$

Class	e^{z_k}	$P(y = k)$
0	3.32	0.086
1	33.1	0.856
2	2.23	0.058
Sum		1.000

Softmax: Loss, Gradient, and Implementation

Categorical cross-entropy loss

Extend the binary cross-entropy to K classes. Let $y_{ik} = 1$ if point i belongs to class k (**one-hot encoding**):

$$J = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} y_{ik} \log P(y = k \mid \mathbf{x}_i)$$

For each example, only the true class k^* contributes:

$$J = -\frac{1}{n}\sum_{i=1}^{n} \log P(y = k_i^* \mid \mathbf{x}_i)$$

Gradient for class k:

$$\nabla_{\mathbf{w}_k} J = \frac{1}{n}\mathbf{X}^T(\hat{\mathbf{p}}_k - \mathbf{y}_k)$$

Same form as binary logistic regression!

Softmax in Python

```
def softmax(Z):
  Z = Z - Z.max(axis=1, keepdims=True)
  E = np.exp(Z)
  return E / E.sum(axis=1, keepdims=True)
def loss(P, Y): # P,Y shape (n,K)
  n = Y.shape[0]
  return -np.sum(Y * np.log(P+1e-9)) / n
def grad(X, P, Y):
  return X.T @ (P - Y) / X.shape[0]
# Training step (W is d×K):
# Z = X @ W     scores (n,K)
# P = softmax(Z)   probs (n,K)

# W -= lr * grad(X, P, Y)
```

Summary: What We Covered

Probability & Loss

- Bernoulli/Binomial model: $P(y \mid p) = p^y(1-p)^{1-y}$
- MLE for coin flip: $\hat{p} = k/n$ (sample proportion)
- Cross-entropy loss = negative log-likelihood of Bernoulli
- Three views: statistics, information theory, optimization

Coefficients & Interpretation

- Log-odds: $\log(\hat{p}/(1-\hat{p})) = \mathbf{w}^T\mathbf{x}$
- Odds ratio for x_j: e^{w_j}
- 1-unit increase in x_j multiplies odds by e^{w_j}

Gradient Derivation

- Chain rule through: loss → sigmoid →

Evaluation Metrics

- Confusion matrix: TP, TN, FP, FN
- Accuracy = $(TP + TN)/n$ — misleading when imbalanced
- Precision = $TP/(TP + FP)$ — quality of positive predictions
- Recall = $TP/(TP + FN)$ — coverage of actual positives
- F_1 = harmonic mean of P & R
- Precision–recall tradeoff via threshold

Extensions

- Imbalanced data: class weights, resampling, threshold tuning
- Multi-class: softmax regression
- Categorical cross-entropy loss; same

Questions?

Next week:

Decision Trees, Random Forests, and Non-linear Classification

Recommended reading:

- Murphy, *Probabilistic Machine Learning: An Introduction*, Chapters 10–11
- Bishop, *Pattern Recognition and Machine Learning*, Chapters 4.3, 8.1
- Hastie et al., *The Elements of Statistical Learning*, Chapters 4.4, 9