Yale University

Logistic Regression and Cross-Entropy

S&DS 265 · Introductory Machine Learning · Lecture 8

Zhuoran Yang

Fall 2026

AI usage: figures were generated, and these slides polished, with the help of AI tools.

Outline

1. Direct Posterior Modeling
2. Bernoulli Likelihood and Cross-Entropy
3. Separable Data and Regularization
4. Gradient Descent Estimation
5. Multiclass Softmax Regression
6. Calibration

Learning Objectives

In this lecture you will learn:

1. Why logistic regression models posterior probabilities through linear log-odds;
2. Why cross-entropy targets the true conditional probability and also follows from likelihood;
3. How the cross-entropy gradient leads to a practical gradient-descent estimator;
4. Why feature maps and regularization control flexibility and complete separation;
5. How vector softmax, multiclass cross-entropy, and stable computation extend the model to many classes.

Outline

Motivating Example: Direct Default Probabilities

The bank again needs a default probability for a new customer. LDA first models how balances and incomes are distributed within each class.

Can we instead model $\mathbb{P}(Y = 1 \mid X = x)$ directly, then choose a review threshold separately from the probability model?

Example and data: ISLP §4.1–4.3 (James, Witten, Hastie, Tibshirani, and Taylor, 2023).

The Default Data

default	student	balance	income
No	No	729.5	44,362
No	Yes	817.2	12,106
No	No	1,073.5	31,767
⋮	⋮	⋮	⋮
No	Yes	200.9	16,863

Source: ISLP §4.3; `statlearning.com`.

Same decision as Lecture 7

10,000 customers, 333 defaults, and a rare-event prediction problem.

Reading One Row: $y_1 = 0$ and x_1 records non-student status, balance 729.5, and income 44,362.

Each row is one customer; `default` is the target.

Problem Setup

Training data. $\mathcal{D}_n = \{(x_i, y_i)\}_{i=1}^n$, drawn independently from an unknown population distribution P.

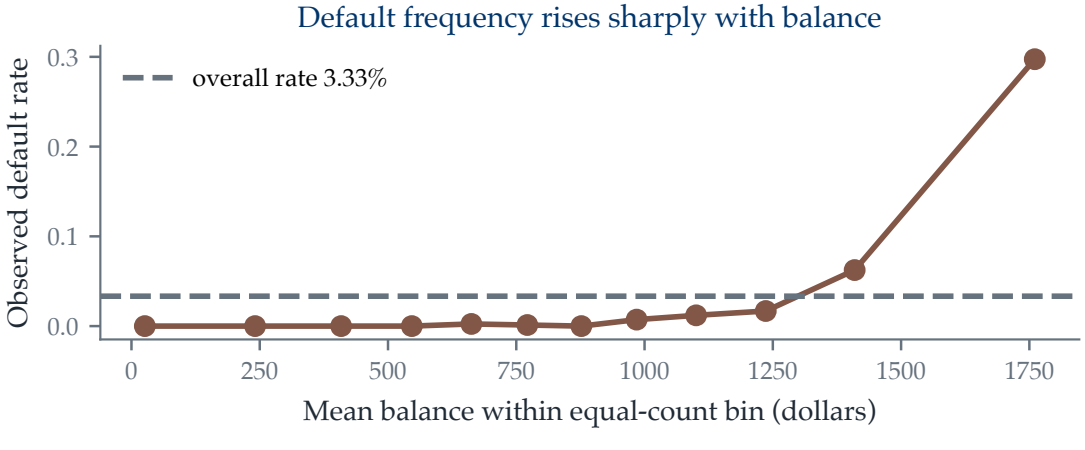
Features. $x_i \in \mathbb{R}^d$ contains measured customer characteristics.

Response. $y_i \in \{0, 1\}$ records non-default or default.

Goal. Estimate $\eta(x^*) = \mathbb{P}(Y = 1 \mid X = x^*)$ for an unseen customer, then choose a review policy reflecting mistake costs.

For `Default`: $n = 10{,}000$, $d = 3$, and $x_i = (\text{student}, \text{balance}, \text{income})^\top$.

Default Frequency and Credit-Card Balance



Course calculation from the ISLP `Default` data.

Recall: LDA Already Gave a Linear Boundary

Lecture 7 modelled both class densities in full. With a shared Σ, everything quadratic cancelled and what survived was

$$\log \frac{\mathbb{P}(Y = 1 \mid x)}{\mathbb{P}(Y = 0 \mid x)} = \delta_1(x) - \delta_0(x) = w^\top x + b.$$

Recall: LDA Already Gave a Linear Boundary

Lecture 7 modelled both class densities in full. With a shared Σ, everything quadratic cancelled and what survived was

$$\log \frac{\mathbb{P}(Y = 1 \mid x)}{\mathbb{P}(Y = 0 \mid x)} = \delta_1(x) - \delta_0(x) = w^\top x + b.$$

After estimating means, covariances and priors, LDA's entire answer was: the log-odds is linear in x.

Recall: LDA Already Gave a Linear Boundary

Lecture 7 modelled both class densities in full. With a shared Σ, everything quadratic cancelled and what survived was

$$\log \frac{\mathbb{P}(Y = 1 \mid x)}{\mathbb{P}(Y = 0 \mid x)} = \delta_1(x) - \delta_0(x) = w^\top x + b.$$

After estimating means, covariances and priors, LDA's entire answer was: the log-odds is linear in x.

The densities were only a means to that end. If the log-odds is what decides the classification, why not model it directly?

The Model Assumption

Logistic regression assumes exactly one thing:

$$\boxed{\log \frac{\mathbb{P}(Y = 1 \mid x)}{\mathbb{P}(Y = 0 \mid x)} = w^\top x + b}$$

The Model Assumption

Logistic regression assumes exactly one thing:

$$\boxed{\log \frac{\mathbb{P}(Y = 1 \mid x)}{\mathbb{P}(Y = 0 \mid x)} = w^\top x + b}$$

- Nothing is assumed about $p(x \mid y)$, about Gaussianity, or about the distribution of X.
- LDA implies it, but so do many other density pairs: the assumption here is strictly weaker.
- Solving for the probability turns it into the sigmoid, so σ is forced rather than posited.

Solving for the Probability Gives the Sigmoid

derive

Write $z = w^\top x + b$ and $p = \mathbb{P}(Y = 1 \mid x)$. The assumption says $\log \frac{p}{1-p} = z$, so

$$\frac{p}{1-p} = e^z \quad \Longrightarrow \quad p = e^z(1-p) \quad \Longrightarrow \quad p(1+e^z) = e^z,$$

hence

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

Solving for the Probability Gives the Sigmoid

derive

Write $z = w^\top x + b$ and $p = \mathbb{P}(Y = 1 \mid x)$. The assumption says $\log \frac{p}{1-p} = z$, so

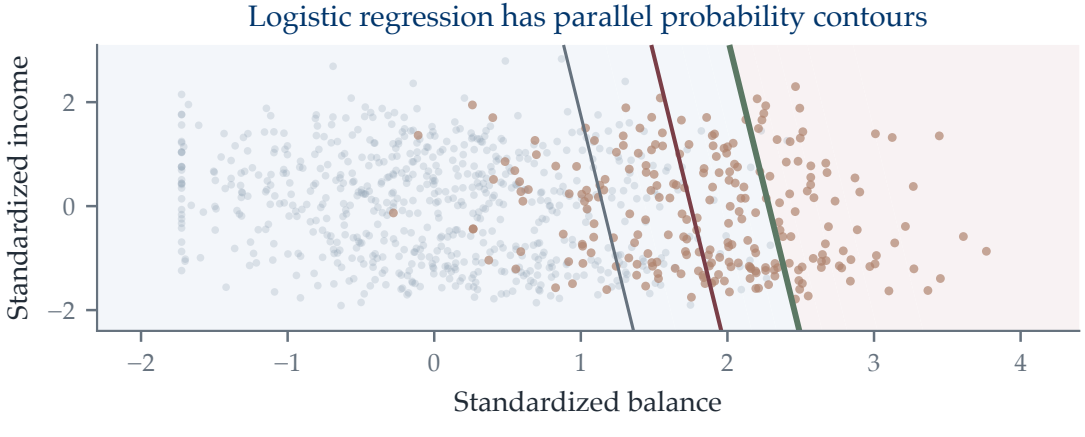
$$\frac{p}{1-p} = e^z \quad \Longrightarrow \quad p = e^z(1-p) \quad \Longrightarrow \quad p(1+e^z) = e^z,$$

hence

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

The probability is nonlinear in x; its log-odds are linear. The sigmoid is not an arbitrary squashing function — it is what the assumption forces.

Fitted Probability Contours on `Default`



Course calculation from the ISLP `Default` training split.

Logistic Regression Is a Parametric Model Class

With a fixed feature map $\phi : \mathbb{R}^d \to \mathbb{R}^m$, define

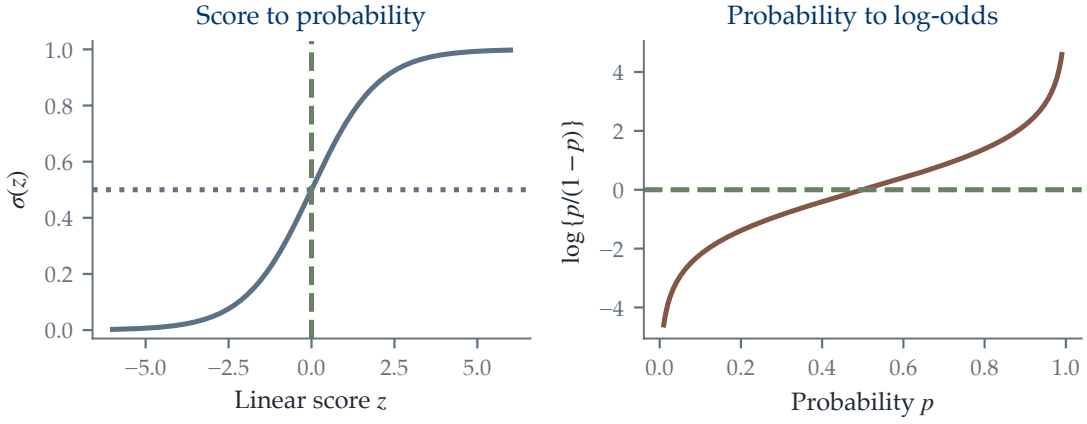
$$\mathcal{F}_{\phi}^{\text{logistic}} = \left\{ x \mapsto \sigma\big(w^\top \phi(x) + b\big) : w \in \mathbb{R}^m,\ b \in \mathbb{R} \right\}.$$

The map ϕ and feature dimension m are chosen before fitting; the finite parameter vector (w, b) is estimated from data.

Linear logistic regression uses $\phi(x) = x$. Polynomial or spline features curve the probability boundary while the model remains linear in the constructed feature space.

Outline

The Sigmoid and Log-Odds Transform



Course calculation of inverse transformations.

Recall: Bernoulli and Binomial

A **Bernoulli**(p) variable takes the value 1 with probability p and 0 with probability $1-p$. Both cases fit in one expression:

$$\mathbb{P}(Y=y) = p^y(1-p)^{1-y}, \qquad y \in \{0,1\},$$

since the exponents switch one factor off. Its mean is $\mathbb{E}Y = p$.

Recall: Bernoulli and Binomial

A **Bernoulli**(p) variable takes the value 1 with probability p and 0 with probability $1-p$. Both cases fit in one expression:

$$\mathbb{P}(Y=y) = p^y(1-p)^{1-y}, \qquad y \in \{0,1\},$$

since the exponents switch one factor off. Its mean is $\mathbb{E}Y = p$.

Adding up n independent Bernoulli(p) variables gives a **Binomial**(n,p) count, with $\mathbb{P}(S=s) = \binom{n}{s}p^s(1-p)^{n-s}$.

Recall: Bernoulli and Binomial

A **Bernoulli**(p) variable takes the value 1 with probability p and 0 with probability $1-p$. Both cases fit in one expression:

$$\mathbb{P}(Y=y) = p^y(1-p)^{1-y}, \qquad y \in \{0,1\},$$

since the exponents switch one factor off. Its mean is $\mathbb{E}Y = p$.

Adding up n independent Bernoulli(p) variables gives a **Binomial**(n,p) count, with $\mathbb{P}(S=s) = \binom{n}{s}p^s(1-p)^{n-s}$.

Here each customer has their own $p_i = \sigma(w^\top x_i + b)$, so the labels are independent Bernoulli variables with different probabilities, not one binomial count.

The Bernoulli Model Covers Both Labels

For $Y_i \in \{0, 1\}$ and $p_i = \mathbb{P}(Y_i = 1 \mid X_i = x_i)$,

$$\mathbb{P}(Y_i = y_i \mid X_i = x_i) = p_i^{y_i}(1 - p_i)^{1-y_i}.$$

Therefore, for conditionally independent observations,

$$L(w, b) = \prod_{i=1}^{n} p_i^{y_i}(1 - p_i)^{1-y_i}, \qquad p_i = \sigma(w^\top x_i + b).$$

The exponents select p_i when $y_i = 1$ and $1 - p_i$ when $y_i = 0$.

Maximum Likelihood Gives Binary Cross-Entropy

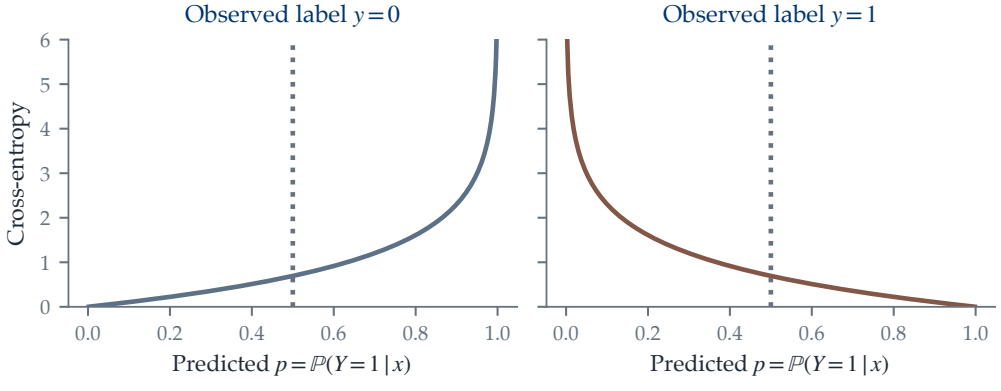
Take the negative logarithm and divide by n:

$$\begin{aligned}\widehat{R}(w,b) &= -\frac{1}{n}\log L(w,b) \\ &= -\frac{1}{n}\sum_{i=1}^{n}\left\{y_i \log p_i + (1-y_i)\log(1-p_i)\right\}.\end{aligned}$$

Equivalently, with $z_i = w^\top x_i + b$,

$$\boxed{\widehat{R}(w,b) = \frac{1}{n}\sum_{i=1}^{n}\left\{\log(1+e^{z_i}) - y_i z_i\right\}.}$$

What the Loss Charges



The cost of reporting p when the label turns out to be 0 or 1. Being right costs almost nothing, being unsure costs $\log 2 \approx 0.69$, and being confidently wrong costs without bound — which is what stops the model from claiming certainty it has not earned.

Course calculation of the Bernoulli negative log-likelihood.

The Model Class Creates Approximation Error

Over all measurable probability functions, cross-entropy is minimized by $\eta(x)$. Logistic regression restricts the search to

$$\mathcal{F}_{\phi}^{\text{logistic}} = \{x \mapsto \sigma(w^{\top}\phi(x) + b)\}.$$

The best function in this class is

$$f_{\phi}^{*} = \underset{f \in \mathcal{F}_{\phi}^{\text{logistic}}}{\arg\min}\, \mathbb{E}[-Y \log f(X) - (1 - Y) \log\{1 - f(X)\}].$$

More observations help estimate f_{ϕ}^{*}, but they cannot remove the gap between f_{ϕ}^{*} and η when the chosen log-odds features are wrong. This is the classification version of the approximation error from regression.

The Same Loss, Written With the Margin

derive

The per-observation loss $\ell_i = -y_i \log p_i - (1 - y_i) \log(1 - p_i)$ has only two cases. With $p_i = \sigma(z_i)$ and $\sigma(-z) = 1 - \sigma(z)$,

$$y_i = 1: \quad \ell_i = -\log \sigma(z_i) = \log(1 + e^{-z_i}), \qquad y_i = 0: \quad \ell_i = \log(1 + e^{z_i}).$$

The Same Loss, Written With the Margin

derive

The per-observation loss $\ell_i = -y_i \log p_i - (1 - y_i) \log(1 - p_i)$ has only two cases. With $p_i = \sigma(z_i)$ and $\sigma(-z) = 1 - \sigma(z)$,

$$y_i = 1: \quad \ell_i = -\log \sigma(z_i) = \log(1 + e^{-z_i}), \qquad y_i = 0: \quad \ell_i = \log(1 + e^{z_i}).$$

Relabel the response as $s_i = 2y_i - 1 \in \{-1, +1\}$. Both lines collapse into one:

$$\boxed{\ell_i = \log(1 + e^{-m_i}), \qquad m_i = s_i z_i = s_i(w^\top x_i + b)}$$

the **logistic loss** as a function of the **margin** m_i.

The Same Loss, Written With the Margin

derive

The per-observation loss $\ell_i = -y_i \log p_i - (1 - y_i) \log(1 - p_i)$ has only two cases. With $p_i = \sigma(z_i)$ and $\sigma(-z) = 1 - \sigma(z)$,

$$y_i = 1: \quad \ell_i = -\log \sigma(z_i) = \log(1 + e^{-z_i}), \qquad y_i = 0: \quad \ell_i = \log(1 + e^{z_i}).$$

Relabel the response as $s_i = 2y_i - 1 \in \{-1, +1\}$. Both lines collapse into one:

$$\boxed{\ell_i = \log(1 + e^{-m_i}), \qquad m_i = s_i z_i = s_i(w^\top x_i + b)}$$

the **logistic loss** as a function of the **margin** m_i.

$m_i > 0$ means the point is on the correct side, and $|m_i|$ says how far. The loss falls as the margin grows and never reaches zero — the fact everything that follows turns on.

Outline

When the Data Can Be Separated

Call the training data **separable** if some direction gets every label right: there is a v with

$$(2y_i - 1)\, v^\top x_i > 0 \qquad \text{for every } i,$$

that is, every point sits on the correct side of the plane $v^\top x = 0$.

When the Data Can Be Separated

Call the training data **separable** if some direction gets every label right: there is a v with

$$(2y_i - 1)\, v^\top x_i > 0 \qquad \text{for every } i,$$

that is, every point sits on the correct side of the plane $v^\top x = 0$.

- Rare when n is large and d small — `Default` is not separable.
- Common when d is large relative to n, and guaranteed once $d \geq n$.
- A feature expansion $\phi(x)$ can create it out of nothing, since it raises d without adding rows.

Separable Data Has No Maximum Likelihood Estimate

derive

In margin form the loss was $\ell_i = \log(1 + e^{-m_i})$ with $m_i = s_i\, w^\top x_i$, decreasing in m_i and never reaching zero.

Separable Data Has No Maximum Likelihood Estimate

derive

In margin form the loss was $\ell_i = \log(1 + e^{-m_i})$ with $m_i = s_i\, w^\top x_i$, decreasing in m_i and never reaching zero.

Walk along the separating direction, $w = cv$ for $c > 0$. Separability says every m_i is positive, so every margin grows with c:

$$\ell_i(cv) = \log(1 + e^{-c\,(2y_i - 1)v^\top x_i}) \;\longrightarrow\; 0 \qquad \text{as } c \to \infty.$$

Separable Data Has No Maximum Likelihood Estimate

derive

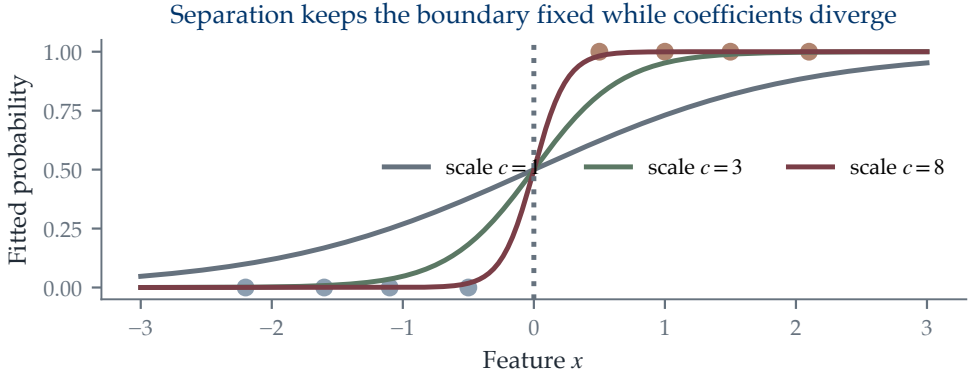
In margin form the loss was $\ell_i = \log(1 + e^{-m_i})$ with $m_i = s_i\, w^\top x_i$, decreasing in m_i and never reaching zero.

Walk along the separating direction, $w = cv$ for $c > 0$. Separability says every m_i is positive, so every margin grows with c:

$$\ell_i(cv) = \log(1 + e^{-c\,(2y_i - 1)v^\top x_i}) \;\longrightarrow\; 0 \qquad \text{as } c \to \infty.$$

The objective decreases forever without reaching a minimum. The supremum of the likelihood is not attained: there is no finite $\widehat{w}$.

The Boundary Stops Moving; the Coefficients Do Not



Three fits along the same direction, at $c = 1, 3, 8$. The boundary is identical in all three — scaling w cannot move it. What changes is confidence: the sigmoid steepens until every training point is assigned probability 0 or 1.

Course illustration on separable data.

Why This Is a Problem

- **No answer to report.** Gradient descent does not converge; it drifts outward, and where it stops depends only on when you stopped it.
- **Absurd confidence.** The model reports probability 1 for points near the boundary, purely because no training point contradicted it. A new point just the other side gets the same certainty.
- **Nothing was learned from the gap.** Every separating plane achieves zero loss in the limit, so the data cannot say which is best — the fit is not identified.

Why This Is a Problem

- **No answer to report.** Gradient descent does not converge; it drifts outward, and where it stops depends only on when you stopped it.
- **Absurd confidence.** The model reports probability 1 for points near the boundary, purely because no training point contradicted it. A new point just the other side gets the same certainty.
- **Nothing was learned from the gap.** Every separating plane achieves zero loss in the limit, so the data cannot say which is best — the fit is not identified.

The cause is that the loss rewards margin without limit. Nothing in it prefers a modest, honest fit.

Regularization Restores a Finite Answer

Add the coefficient penalty from Lecture 4:

$$\min_{w,b} \; -\frac{1}{n}\sum_{i=1}^{n}\{y_i \log p_i + (1-y_i)\log(1-p_i)\} + \frac{\lambda}{2}\|w\|_2^2$$

Regularization Restores a Finite Answer

Add the coefficient penalty from Lecture 4:

$$\min_{w,b} \; -\frac{1}{n}\sum_{i=1}^{n}\{y_i \log p_i + (1-y_i)\log(1-p_i)\} + \frac{\lambda}{2}\|w\|_2^2$$

Now walking out along cv drives the loss to 0 but the penalty to ∞, so the objective eventually increases. It is strictly convex in w, so the minimizer exists and is unique.

Regularization Restores a Finite Answer

Add the coefficient penalty from Lecture 4:

$$\min_{w,b} \; -\frac{1}{n}\sum_{i=1}^{n}\{y_i \log p_i + (1-y_i)\log(1-p_i)\} + \frac{\lambda}{2}\|w\|_2^2$$

Now walking out along cv drives the loss to 0 but the penalty to ∞, so the objective eventually increases. It is strictly convex in w, so the minimizer exists and is unique.

- $\|w\|_1$ instead gives sparse logistic regression, as the lasso did for least squares.
- The intercept b is left unpenalized, and λ is chosen by cross-validation.

Outline

The Gradient Is a Residual Times a Feature

derive

Write one observation's loss in terms of its score $z_i = w^\top x_i + b$ and probability $p_i = \sigma(z_i)$:

$$\ell_i = -y_i \log p_i - (1 - y_i) \log(1 - p_i), \qquad \widehat{R}_n = \frac{1}{n} \sum_{i=1}^{n} \ell_i.$$

The Gradient Is a Residual Times a Feature

derive

Write one observation's loss in terms of its score $z_i = w^\top x_i + b$ and probability $p_i = \sigma(z_i)$:

$$\ell_i = -y_i \log p_i - (1 - y_i) \log(1 - p_i), \qquad \widehat{R}_n = \frac{1}{n} \sum_{i=1}^{n} \ell_i.$$

The chain rule takes it in two steps:

$$\frac{\partial \ell_i}{\partial w} = \frac{\partial \ell_i}{\partial p_i} \cdot \frac{dp_i}{dz_i} \cdot \frac{\partial z_i}{\partial w}, \qquad \frac{dp_i}{dz_i} = p_i(1 - p_i), \qquad \frac{\partial z_i}{\partial w} = x_i.$$

The Gradient Is a Residual Times a Feature

derive

Write one observation's loss in terms of its score $z_i = w^\top x_i + b$ and probability $p_i = \sigma(z_i)$:

$$\ell_i = -y_i \log p_i - (1 - y_i) \log(1 - p_i), \qquad \widehat{R}_n = \frac{1}{n} \sum_{i=1}^{n} \ell_i.$$

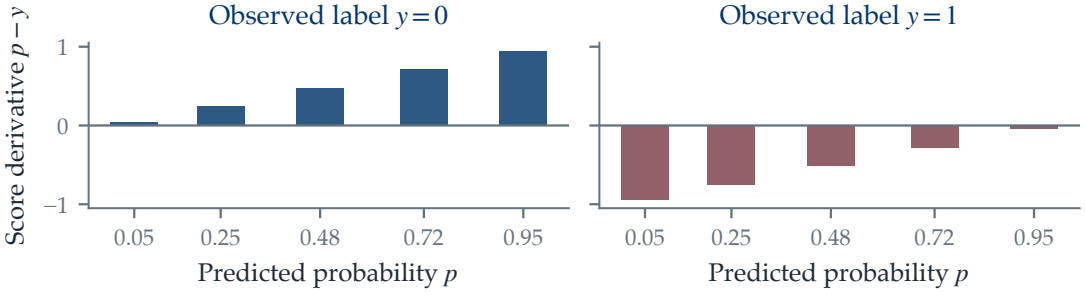
The chain rule takes it in two steps:

$$\frac{\partial \ell_i}{\partial w} = \frac{\partial \ell_i}{\partial p_i} \cdot \frac{dp_i}{dz_i} \cdot \frac{\partial z_i}{\partial w}, \qquad \frac{dp_i}{dz_i} = p_i(1 - p_i), \qquad \frac{\partial z_i}{\partial w} = x_i.$$

The first two cancel, leaving the **probability residual**:

$$\frac{\partial \ell_i}{\partial z_i} = \boxed{p_i - y_i} \quad \Longrightarrow \quad \nabla_w \widehat{R}_n = \frac{1}{n} \sum_{i=1}^{n} (p_i - y_i)\, x_i.$$

The Residual Says Which Way and How Hard



The residual $p - y$ for each label. A confident correct prediction gives almost zero, so that row barely moves the coefficients; a confident wrong one gives nearly ± 1 and pulls hardest. The sign says which direction.

Course calculation of the score derivative.

In Matrix Form

Stacking the rows $x_i^\top$ into $X \in \mathbb{R}^{n \times d}$ and the residuals into $p - y \in \mathbb{R}^n$,

$$\boxed{\nabla_w \widehat{R}_n = \frac{1}{n} X^\top (p - y)}, \qquad \boxed{\frac{\partial \widehat{R}_n}{\partial b} = \frac{1}{n} \mathbf{1}^\top (p - y)}.$$

In Matrix Form

Stacking the rows $x_i^\top$ into $X \in \mathbb{R}^{n \times d}$ and the residuals into $p - y \in \mathbb{R}^n$,

$$\boxed{\nabla_w \widehat{R}_n = \frac{1}{n} X^\top (p - y)}, \qquad \boxed{\frac{\partial \widehat{R}_n}{\partial b} = \frac{1}{n} \mathbf{1}^\top (p - y)}.$$

- One matrix–vector product per step, exactly as in Lecture 5: the cost is $O(nd)$ and a minibatch replaces X by a slice of its rows.
- Compare least squares, where the residual was $X\theta - y$. Here it is $\sigma(Xw + b) - y$: the only change is the sigmoid.

The Logistic Objective Is Convex

For one score z, binary cross-entropy can be written as

$$\ell(z, y) = \log(1 + e^z) - yz.$$

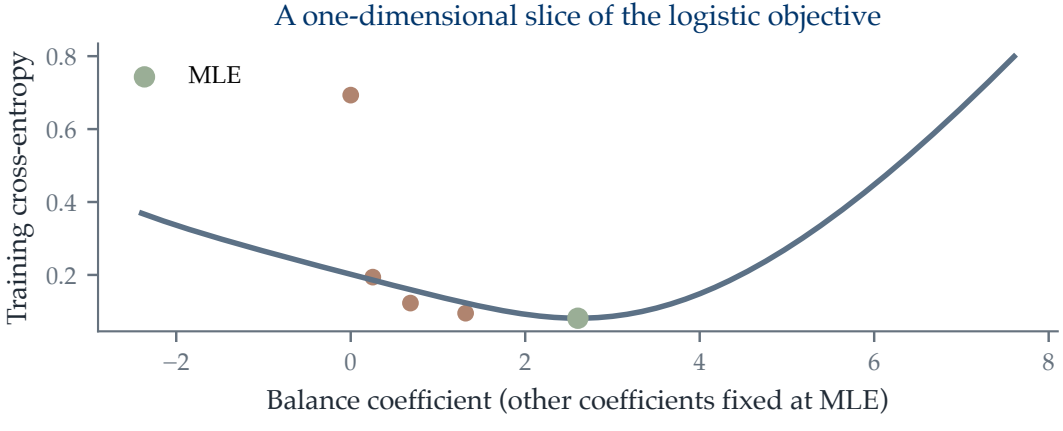
Its curvature is nonnegative:

$$\frac{\partial^2 \ell}{\partial z^2} = p(1 - p) \geq 0.$$

Since $z = w^\top x + b$ is affine in (w, b), averaging these losses preserves convexity. Thus gradient descent cannot be trapped in a worse local minimum.

Complete separation is different: the unregularized infimum may occur only as $\|w\| \to \infty$, even though the objective is convex.

The Objective Is Bowl-Shaped



A slice through the logistic objective. Convexity means gradient descent cannot be trapped: every stationary point is a global minimum, exactly as for least squares in Lecture 3.

Course calculation on the ISLP `Default` data.

Gradient Descent Fits the Coefficients

Starting from (w_0, b_0), repeat

$$\boxed{\begin{aligned} w_{t+1} &= w_t - \alpha_t \frac{1}{n} X^\top (p_t - y), \\ b_{t+1} &= b_t - \alpha_t \frac{1}{n} \mathbf{1}^\top (p_t - y), \end{aligned}}$$

where $(p_t)_i = \sigma(w_t^\top x_i + b_t)$ and $\alpha_t > 0$ is a step size.

- Each step costs $O(nd)$ for a dense full batch.
- A penalty adds its derivative; for ridge, add λw_t.
- Minibatches give stochastic gradients when n is large.

The Logistic Regression Fitting Workflow

1. Choose and standardize the feature map $\phi(x)$;
2. Choose a penalty $\Omega(w)$ and candidate values of λ;
3. Minimize regularized empirical cross-entropy with gradient descent or a standard numerical solver;
4. Select λ using validation data, then refit the selected model;
5. Evaluate held-out probability quality before selecting a deployment threshold from costs.

The optimizer estimates (w, b); validation chooses modeling complexity; the threshold chooses an action. These are separate steps.

Outline

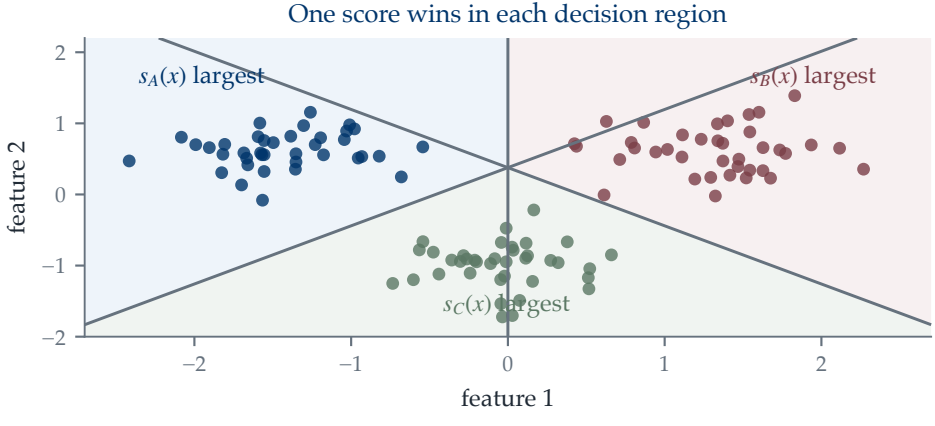
Why Do We Need a Vector of Probabilities?

Binary logistic regression answers one question: "class 1 or class 0?" Many tasks require one choice among $K > 2$ unordered classes.

An image may be a cat, dog, or bird; a language model must choose a distribution over every possible next token.

Multinomial logistic regression: ISLP §4.3.5. Softmax and stable cross-entropy: Dive into Deep Learning, Chapter 4.

Three Scores Partition the Feature Space



A binary probability is no longer enough. At every input x, the model must compare one score per class and return probabilities summing to one.

Construction follows the multiclass examples in ISLP §4.2 and Dive into Deep Learning, Chapter 4.

The Target Is a Conditional Probability Vector

For $Y \in \{1, \dots, K\}$, define

$$\eta(x) = \begin{bmatrix} \mathbb{P}(Y = 1 \mid X = x) \\ \vdots \\ \mathbb{P}(Y = K \mid X = x) \end{bmatrix} \in \Delta^{K-1}, \qquad \Delta^{K-1} = \left\{ q \in \mathbb{R}^K : q_k \geq 0,\ \sum_k q_k = 1 \right\}.$$

A label $y_i = k$ can be represented by the **one-hot vector** $e_k \in \{0, 1\}^K$.

Class numbers are names, not measurements: coding cat, dog, and bird as $1, 2, 3$ does not make dog halfway between cat and bird.

The Multiclass Model Assumption

Binary logistic assumed one log-odds was linear. With K classes there is a log-odds for every pair, and the assumption is that all of them are:

$$\log \frac{\mathbb{P}(Y = k \mid x)}{\mathbb{P}(Y = l \mid x)} = (w_k - w_l)^\top \phi(x) + (b_k - b_l)$$

The Multiclass Model Assumption

Binary logistic assumed one log-odds was linear. With K classes there is a log-odds for every pair, and the assumption is that all of them are:

$$\log \frac{\mathbb{P}(Y = k \mid x)}{\mathbb{P}(Y = l \mid x)} = (w_k - w_l)^\top \phi(x) + (b_k - b_l)$$

- Equivalently: fix any class as a reference and model each $\log(\mathbb{P}(Y = k \mid x)/\mathbb{P}(Y = K \mid x))$ as linear.
- Only the differences $w_k - w_l$ are assumed to be anything, so adding a common vector to every w_k changes nothing.
- At $K = 2$ this is exactly the binary assumption.

The Linear Model Produces One Score per Class

Choose a feature map $\phi(x) \in \mathbb{R}^m$. For class k, define a logit

$$s_k(x) = w_k^\top \phi(x) + b_k.$$

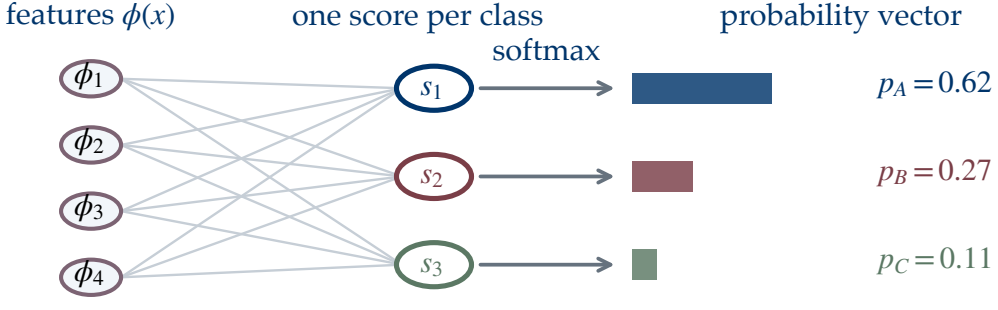
Stack the class parameters as rows of $W \in \mathbb{R}^{K \times m}$ and let $b \in \mathbb{R}^K$:

$$\boxed{s(x) = W\phi(x) + b \in \mathbb{R}^K.}$$

These **logits** are unrestricted comparison scores, not yet probabilities. The model now needs a vector-valued link

$$\mathbb{R}^K \longrightarrow \Delta^{K-1}.$$

A Linear Layer Produces K Logits



Each class has its own affine score. Softmax couples all K scores into one probability distribution.

Adapted conceptually from Dive into Deep Learning's softmax-regression diagram (CC BY-SA 4.0).

Softmax Converts Logits into Probabilities

For $s = (s_1, \dots, s_K)^\top \in \mathbb{R}^K$, define

$$\boxed{\operatorname{softmax}(s)_k = p_k = \frac{e^{s_k}}{\sum_{j=1}^{K} e^{s_j}}}$$

- Positive and summing to one, so it lands in the simplex;
- Equal logits give $p_k = 1/K$, and adding a constant to every logit changes nothing.

Softmax Converts Logits into Probabilities

For $s = (s_1, \dots, s_K)^\top \in \mathbb{R}^K$, define

$$\boxed{\operatorname{softmax}(s)_k = p_k = \frac{e^{s_k}}{\sum_{j=1}^{K} e^{s_j}}}$$

- Positive and summing to one, so it lands in the simplex;
- Equal logits give $p_k = 1/K$, and adding a constant to every logit changes nothing.

Softmax applied to a linear score is exactly the assumption: take logs of p_k/p_l and the normalizer cancels, leaving $(w_k - w_l)^\top \phi(x) + (b_k - b_l)$. The model is now fully specified.

Aside: Temperature Rescales Confidence

Divide every logit by a constant $T > 0$ before the softmax:

$$p_k(T) = \frac{e^{s_k/T}}{\sum_j e^{s_j/T}}.$$

Aside: Temperature Rescales Confidence

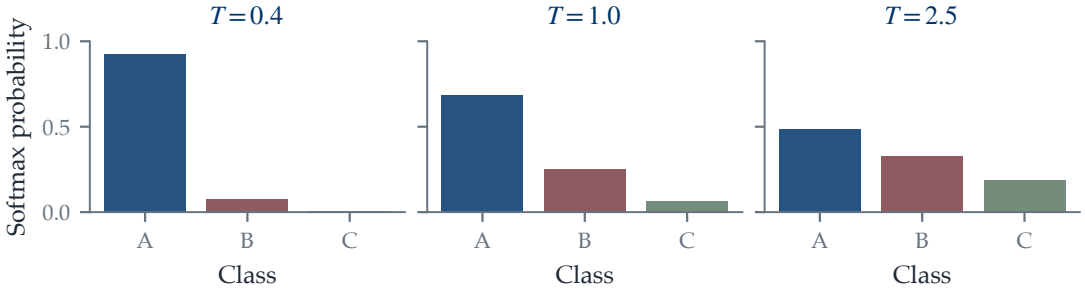
Divide every logit by a constant $T > 0$ before the softmax:

$$p_k(T) = \frac{e^{s_k/T}}{\sum_j e^{s_j/T}}.$$

- $T = 1$ is the fitted model; $T \to 0$ sends all mass to the largest logit, $T \to \infty$ flattens toward uniform $1/K$.
- Dividing by $T > 0$ preserves the order of the logits, so the arg max never moves: temperature changes confidence, not the decision.

Used for sampling from a language model, and for **temperature scaling**: fitting one T on held-out data to repair an over-confident model without refitting it.

Scaling the Logits Sharpens the Distribution



Softmax applied to the same logits divided by a temperature. Large temperature flattens toward the uniform distribution; small temperature concentrates on the largest logit. The ranking never changes.

The same device controls sampling from a language model.

Recall: Categorical and Multinomial

A **categorical** variable takes one of K values with probabilities $\pi_1, \dots, \pi_K$ summing to one. Writing the label as a **one-hot** vector $y \in \{0,1\}^K$ with a single 1,

$$\mathbb{P}(Y = y) = \prod_{k=1}^{K} \pi_k^{y_k},$$

the same trick as Bernoulli: the exponents switch off every factor but one.

Recall: Categorical and Multinomial

A **categorical** variable takes one of K values with probabilities $\pi_1, \dots, \pi_K$ summing to one. Writing the label as a **one-hot** vector $y \in \{0,1\}^K$ with a single 1,

$$\mathbb{P}(Y = y) = \prod_{k=1}^{K} \pi_k^{y_k},$$

the same trick as Bernoulli: the exponents switch off every factor but one.

Counting n independent draws gives a **Multinomial**(n, π) vector of counts. For $K = 2$ both reduce to Bernoulli and binomial.

Recall: Categorical and Multinomial

A **categorical** variable takes one of K values with probabilities $\pi_1, \dots, \pi_K$ summing to one. Writing the label as a **one-hot** vector $y \in \{0,1\}^K$ with a single 1,

$$\mathbb{P}(Y = y) = \prod_{k=1}^{K} \pi_k^{y_k},$$

the same trick as Bernoulli: the exponents switch off every factor but one.

Counting n independent draws gives a **Multinomial**(n, π) vector of counts. For $K = 2$ both reduce to Bernoulli and binomial.

As before, each row has its own probability vector $\pi = p(x_i)$, so the labels are independent categorical draws with different probabilities.

Multinomial Likelihood Gives Cross-Entropy

Let $y_{ik} = \mathbf{1}\{y_i = k\}$ and $p_{ik} = \text{softmax}(s_i)_k$. The categorical likelihood is

$$L(W, b) = \prod_{i=1}^{n} \prod_{k=1}^{K} p_{ik}^{y_{ik}}.$$

Taking the negative log and averaging gives

$$\boxed{\widehat{R}(W, b) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} y_{ik} \log p_{ik} = -\frac{1}{n} \sum_{i=1}^{n} \log p_{i,y_i}.}$$

Cross-entropy therefore asks one concrete question for each observation: how much probability did the model assign to the observed class?

The Softmax Gradient Is Again $p - y$

Differentiating the multiclass cross-entropy through the softmax produces the same cancellation as the binary case:

$$\nabla_z \ell = \boxed{p - y} \qquad \nabla_W \widehat{R}_n = \frac{1}{n} \sum_{i=1}^{n} (p_i - y_i) x_i^\top,$$

with p the predicted probability vector and y the one-hot label.

The Softmax Gradient Is Again $p - y$

Differentiating the multiclass cross-entropy through the softmax produces the same cancellation as the binary case:

$$\nabla_z \ell = \boxed{p - y} \qquad \nabla_W \widehat{R}_n = \frac{1}{n} \sum_{i=1}^{n} (p_i - y_i) x_i^\top,$$

with p the predicted probability vector and y the one-hot label.

Binary and multiclass, least squares and logistic: every model in this course so far has a gradient of the form residual times feature. That is the pattern backpropagation will generalize.

Source: Derived in the appendix.

Stable Softmax Subtracts the Largest Logit

Directly evaluating e^{s_k} can overflow. Let $m = \max_j s_j$. Since softmax is unchanged by a common shift,

$$\boxed{p_k = \frac{e^{s_k - m}}{\sum_j e^{s_j - m}}.}$$

Now every exponent satisfies $s_k - m \leq 0$, so every numerator is at most 1.

For $s = (1000, 999, 998)$,

$(e^{1000}, e^{999}, e^{998})$ overflows, (e^0, e^{-1}, e^{-2}) is safe.

Both represent exactly the same softmax probabilities.

Stable Cross-Entropy Uses Log-Sum-Exp

Do not compute p_y and then take $-\log p_y$. Compute from logits:

$$\boxed{\ell(s, y) = -s_y + m + \log \sum_j e^{s_j - m}}, \qquad m = \max_j s_j.$$

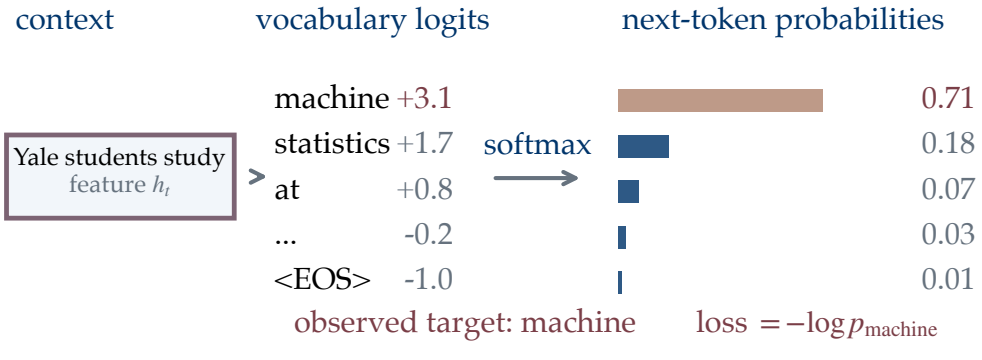
This avoids overflow and avoids taking $\log 0$ after probability underflow.

Framework rule

Pass raw logits to a fused cross-entropy routine. In PyTorch:
`torch.nn.functional.cross_entropy(logits, targets)`.

Source: Stable softmax derivation follows Dive into Deep Learning, "Softmax Revisited."

Next-Token Prediction Is Multiclass Classification



The context model supplies one feature vector. A vocabulary-sized linear layer produces one logit per token, and softmax turns them into the next-token distribution.

This bridge is developed fully in Lecture 21.

Outline

Calibration Compares Confidence with Frequency

A model is **calibrated** if its stated probabilities match observed frequencies:

$$\mathbb{P}(Y = 1 \mid p(X) = q) = q.$$

Among the customers it calls 10% risks, about 10% should default.

Calibration Compares Confidence with Frequency

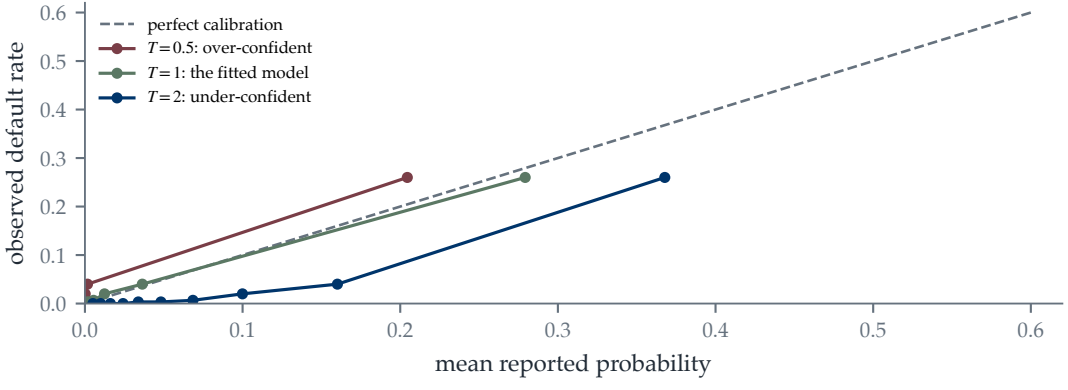
A model is **calibrated** if its stated probabilities match observed frequencies:

$$\mathbb{P}(Y = 1 \mid p(X) = q) = q.$$

Among the customers it calls 10% risks, about 10% should default.

- Check it with a **reliability curve**: bin by predicted probability and plot observed frequency against mean prediction.
- Calibration is about the numbers; accuracy is about the ranking. A model can be excellent at one and hopeless at the other.

The Same Model at Three Temperatures



Dividing the logits by T cannot reorder anything, so all three curves come from a model with identical AUC (0.941807) and identical accuracy (97.47%). Only the honesty of the numbers changes.

Course calculation on a held-out 30% split of the ISLP `Default` data.

Accuracy Cannot See Miscalibration

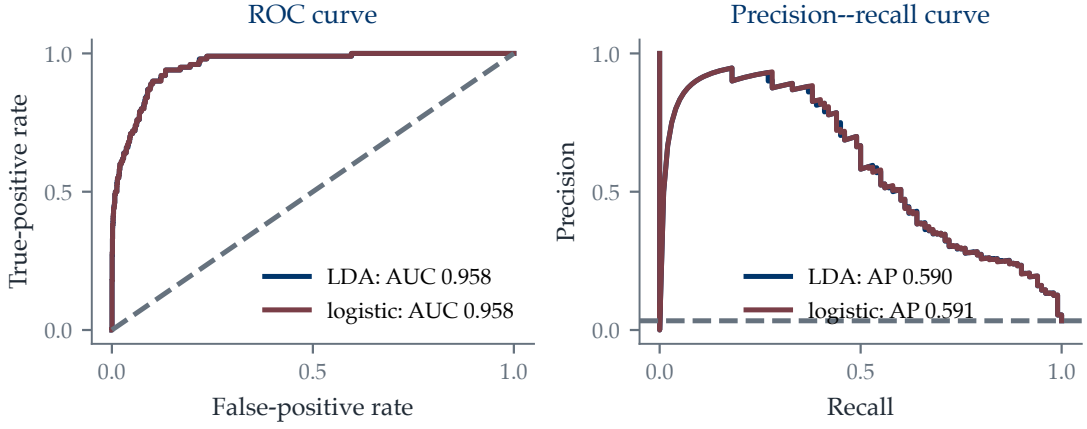
	AUC	accuracy	calibration error
$T = 0.5$, over-confident	0.941807	97.47%	0.013
$T = 1$, the fitted model	0.941807	97.47%	0.004
$T = 2$, under-confident	0.941807	97.47%	0.050

Accuracy Cannot See Miscalibration

	AUC	accuracy	calibration error
$T = 0.5$, over-confident	0.941807	97.47%	0.013
$T = 1$, the fitted model	0.941807	97.47%	0.004
$T = 2$, under-confident	0.941807	97.47%	0.050

- The first two columns are identical to every digit shown, because a monotone rescaling changes no ranking and no decision.
- Maximum likelihood is what puts $T = 1$ on the diagonal: cross-entropy is minimized by the true probability, so fitting it is fitting calibration.
- To repair a miscalibrated model, fit one T on held-out data. It cannot harm accuracy.

LDA and Logistic Regression on the Same Split



Course calculation on the identical stratified split of the ISLP `Default` data.

Summary: Multiclass Decisions and Evaluation

$$s(x) = W\phi(x) + b, \qquad p_k(x) = \frac{e^{s_k(x)}}{\sum_j e^{s_j(x)}}, \qquad \widehat{R} = -\frac{1}{n}\sum_i \log p_{i,y_i}.$$

- One linear score per class defines the model class; only score differences affect probabilities.
- Multiclass cross-entropy is negative log likelihood, and its logit gradient is $p - y$.
- Stable implementations subtract the maximum logit and fuse softmax with log-sum-exp cross-entropy.
- A language model uses the same output model with tokens as classes.

Next Lecture

Logistic regression turns a linear score into a probability. When classes are nearly separable, a different question becomes natural: which separating boundary leaves the most room around the training observations?

Lecture 9 — Margin Classification replaces probability modeling with a geometric objective, then extends the hard-margin idea through soft margins, feature maps, and kernels.

Reading for this lecture: ISLP Chapter 4; D2L Chapter 4.

Appendix

Derivations skipped in the lecture hour, kept for completeness.

A. **Why cross-entropy is the right loss.** It is minimized, in the population, exactly at the true conditional probability.

B. **Where $p - y$ comes from.** The sigmoid derivative and the logarithm cancel, in both the binary and the multiclass case.

C. **Softmax facts.** Only score differences matter, and two classes reduce to the binary model.

A.1 Cross-Entropy Targets the True Posterior

derive

At a fixed x, let $\eta = \mathbb{P}(Y = 1 \mid X = x)$ and suppose we report $q \in (0, 1)$. The conditional expected cross-entropy is

$$r_x(q) = -\eta \log q - (1 - \eta) \log(1 - q).$$

Differentiating,

$$r'_x(q) = -\frac{\eta}{q} + \frac{1 - \eta}{1 - q} = \frac{q - \eta}{q(1 - q)}.$$

Hence the unique stationary point is

$$\boxed{q^*(x) = \eta(x) = \mathbb{P}(Y = 1 \mid X = x)},$$

and $r''_x(q) > 0$ there. Cross-entropy therefore rewards the true conditional probability in the population.

B.1 Where the Residual $p - y$ Comes From

derive

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

$$\begin{aligned}
\frac{dp}{dz} &= (1 + e^{-z})^{-2} e^{-z} = p(1 - p), \\
\frac{\partial \ell}{\partial z} &= \left(-\frac{y}{p} + \frac{1 - y}{1 - p} \right) \frac{dp}{dz} \\
&= -y(1 - p) + (1 - y)p \\
&= \boxed{p - y}.
\end{aligned}$$

The sigmoid derivative and the logarithms cancel, leaving the simple probability residual $p - y$.

B.2 The Softmax Gradient

derive

For one observation with logits s and label vector y, write

$$\ell(s, y) = -\sum_k y_k \log p_k = -y^\top s + \log \sum_j e^{s_j}.$$

Differentiating each logit,

$$\boxed{\frac{\partial \ell}{\partial s_k} = p_k - y_k}, \qquad \boxed{\nabla_W \ell = (p - y)\phi(x)^\top}.$$

The true class has $y_k = 1$, so its logit is pushed upward; every other class has $y_k = 0$, so its logit is pushed downward in proportion to its current probability. Gradient descent then averages these contributions over data.

C.1 Only Score Differences Matter

derive

Adding the same constant c to every logit leaves the probabilities fixed:

$$\text{softmax}(s + c\mathbf{1})_k = \frac{e^{s_k+c}}{\sum_j e^{s_j+c}} = \frac{e^c e^{s_k}}{e^c \sum_j e^{s_j}} = \text{softmax}(s)_k.$$

Equivalently, pairwise log-odds are score differences:

$$\boxed{\log \frac{p_k(x)}{p_\ell(x)} = s_k(x) - s_\ell(x).}$$

Thus W, b are not uniquely identified without a convention. We may set one class's coefficients to zero, or use the symmetric parameterization and let regularization select a representative.

C.2 Two-Class Softmax Is Binary Logistic

derive

With $K = 2$,

$$p_1 = \frac{e^{s_1}}{e^{s_1} + e^{s_2}} = \frac{1}{1 + e^{-(s_1 - s_2)}}$$
$$= \sigma(s_1 - s_2), \qquad p_2 = 1 - p_1.$$

If $s_k = w_k^\top \phi(x) + b_k$, then

$$s_1 - s_2 = (w_1 - w_2)^\top \phi(x) + (b_1 - b_2),$$

exactly the binary logistic model.

Softmax is not a separate trick: it is the symmetric K-class extension of the sigmoid model.

C.3 Softmax Follows From the Assumption

derive

Take class K as reference and write $z_k = s_k(x) - s_K(x)$, so $z_K = 0$. The assumption says $\mathbb{P}(Y = k \mid x) = \mathbb{P}(Y = K \mid x)\, e^{z_k}$. Summing over k,

$$1 = \mathbb{P}(Y = K \mid x) \sum_{j=1}^{K} e^{z_j} \quad \Longrightarrow \quad \mathbb{P}(Y = K \mid x) = \frac{1}{\sum_j e^{z_j}}.$$

C.3 Softmax Follows From the Assumption

derive

Take class K as reference and write $z_k = s_k(x) - s_K(x)$, so $z_K = 0$. The assumption says $\mathbb{P}(Y = k \mid x) = \mathbb{P}(Y = K \mid x)\, e^{z_k}$. Summing over k,

$$1 = \mathbb{P}(Y = K \mid x) \sum_{j=1}^{K} e^{z_j} \quad \Longrightarrow \quad \mathbb{P}(Y = K \mid x) = \frac{1}{\sum_j e^{z_j}}.$$

Substituting back and cancelling the common factor e^{-s_K},

$$\boxed{\mathbb{P}(Y = k \mid x) = \frac{e^{s_k(x)}}{\sum_{j=1}^{K} e^{s_j(x)}} = \mathrm{softmax}(s)_k}$$

so softmax is forced by the assumption, exactly as the sigmoid was in the binary case.