Logistic Regression - Derivations & Proofs

Paul F. Roysdon, Ph.D.

Contents

1 Mathematical Derivations & Proofs

1.1 Introduction

Logistic regression is a probabilistic model for binary classification that maps a linear score to a probability via the logistic (sigmoid) function. We derive the model from first principles: data and notation, model formulation, maximum likelihood and cross-entropy, gradients and Hessian, Newton/ Iteratively Reweighted Least Squares (IRLS), regularization (and MAP view), numerical stability via the Log-Sum-Exp trick, multiclass (softmax) extension, and a concise algorithm. Dimensions and properties are made explicit, and all vectors/matrices use the boldface convention.

1.2 Data and Notation

Given a dataset

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}, \qquad \mathbf{x}_i \in \mathbb{R}^d, \quad y_i \in \{0, 1\},$$

with

$$n = \text{number of samples}, \qquad d = \text{number of features}.$$

A linear score with intercept is

$$z_i \;=\; \mathbf{w}^\top \mathbf{x}_i + b, \qquad \mathbf{w} \in \mathbb{R}^d, \quad b \in \mathbb{R}.$$

For GLM notation we often augment features and parameters:

$$\tilde{\mathbf{x}}_i = \begin{bmatrix} 1 \\ \mathbf{x}_i \end{bmatrix} \in \mathbb{R}^{d+1}, \quad \boldsymbol{\beta} = \begin{bmatrix} b \\ \mathbf{w} \end{bmatrix} \in \mathbb{R}^{d+1}, \quad \eta_i = \tilde{\mathbf{x}}_i^\top \boldsymbol{\beta} \equiv z_i.$$

1.3 Model Formulation

The logistic (sigmoid) function is

$$\sigma(t) \;=\; \frac{1}{1+e^{-t}}, \qquad \sigma'(t) = \sigma(t)\big(1-\sigma(t)\big).$$

The model predicts the Bernoulli parameter

$$p_i \;\triangleq\; \Pr(Y=1 \mid \mathbf{x}_i; \mathbf{w}, b) \;=\; \sigma(z_i) \;=\; \sigma(\tilde{\mathbf{x}}_i^\top \boldsymbol{\beta}) \in (0,1),$$

and the sample likelihood is

$$\Pr(y_i \mid \mathbf{x}_i; \mathbf{w}, b) \;=\; p_i^{\,y_i}\,(1-p_i)^{\,1-y_i}.$$

1.4 Maximum Likelihood and Cross-Entropy

Assuming that the samples are independently distributed, the likelihood of the entire dataset is:

$$L(\mathbf{w}, b) = \prod_{i=1}^{n} P(y_i \mid \mathbf{x}_i; \mathbf{w}, b) = \prod_{i=1}^{n} \sigma(z_i)^{y_i} \Big(1-\sigma(z_i)\Big)^{1-y_i}.$$

It is often more convenient to maximize the log-likelihood:

$$\begin{aligned} \ell(\mathbf{w}, b) &= \sum_{i=1}^{n} \Big[y_i \log p_i + (1-y_i)\log(1-p_i) \Big] \qquad (1) \\ &= \sum_{i=1}^{n} \Big[y_i z_i - \log(1+e^{z_i}) \Big] = \sum_{i=1}^{n} \Big[y_i \eta_i - \log(1+e^{\eta_i}) \Big]. \end{aligned}$$

The cost function (or loss function) is defined as the negative log-likelihood (also known as the **cross-entropy loss**). Maximizing ℓ is equivalent to minimizing the convex *cross-entropy* (negative log-likelihood)

$$\begin{aligned} J(\mathbf{w}, b) \;=\; -\,\ell(\mathbf{w}, b) &= \sum_{i=1}^{n} \Big[\log(1+e^{z_i}) - y_i z_i \Big] \\ &= \sum_{i=1}^{n} \Big[-y_i \log p_i - (1-y_i)\log(1-p_i) \Big]. \qquad (2) \end{aligned}$$

1.5 Gradients, Hessian, and Convexity

Gradient with Respect to w

Recall that for each sample:

$$z_i = \mathbf{w}^\top \mathbf{x}_i + b,$$

and the sigmoid function is:

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

Its derivative with respect to z_i is:

$$\sigma'(z_i) = \sigma(z_i)(1-\sigma(z_i)).$$

The gradient of the cost function with respect to $\mathbf{w}$ is given by:

$$\frac{\partial J}{\partial \mathbf{w}} = -\sum_{i=1}^{n} \left[\frac{y_i}{\sigma(z_i)} \sigma'(z_i)\mathbf{x}_i - \frac{1-y_i}{1-\sigma(z_i)} \sigma'(z_i)\mathbf{x}_i \right].$$

Substitute $\sigma'(z_i) = \sigma(z_i)(1 - \sigma(z_i))$ to obtain:

$$\frac{\partial J}{\partial \mathbf{w}} = -\sum_{i=1}^{n} \left[y_i(1 - \sigma(z_i))\mathbf{x}_i - (1 - y_i)\sigma(z_i)\mathbf{x}_i \right].$$

After algebraic manipulation, this simplifies to:

$$\frac{\partial J}{\partial \mathbf{w}} = \sum_{i=1}^{n} \left[\sigma(z_i) - y_i \right] \mathbf{x}_i.$$

Dimensions: Each $\mathbf{x}_i$ is $d \times 1$ and the sum is over $i = 1, \ldots, n$; hence, $\frac{\partial J}{\partial \mathbf{w}} \in \mathbb{R}^d$.

Gradient with Respect to b

Similarly, the derivative of the cost function with respect to b is:

$$\frac{\partial J}{\partial b} = \sum_{i=1}^{n} \left[\sigma(z_i) - y_i \right].$$

Dimensions: Since b is a scalar, $\frac{\partial J}{\partial b} \in \mathbb{R}$.

Reparameterization

Using $p_i = \sigma(z_i)$ and $\partial p_i / \partial z_i = p_i(1 - p_i)$, we obtain

$$\frac{\partial J}{\partial \mathbf{w}} = \sum_{i=1}^{n} (p_i - y_i)\, \mathbf{x}_i, \qquad \frac{\partial J}{\partial b} = \sum_{i=1}^{n} (p_i - y_i).$$

Stacking $\tilde{\mathbf{x}}_i^\top$ as rows of $\mathbf{X} \in \mathbb{R}^{n \times (d+1)}$, define

$$\mathbf{p} = (p_1, \ldots, p_n)^\top, \quad \mathbf{y} = (y_1, \ldots, y_n)^\top, \quad \mathbf{W} = \operatorname{diag}\big(p_i(1 - p_i)\big) \succeq \mathbf{0}.$$

Then with $\boldsymbol{\beta} = [b; \mathbf{w}]$,

$$\nabla J(\boldsymbol{\beta}) = \mathbf{X}^\top(\mathbf{p} - \mathbf{y}), \tag{3}$$

$$\nabla^2 J(\boldsymbol{\beta}) = \mathbf{X}^\top \mathbf{W} \mathbf{X} \;\succeq\; \mathbf{0}, \tag{4}$$

so J is convex (strictly convex if $\mathbf{X}$ has full column rank and $0 < p_i < 1$).

1.6 Newton–Raphson and IRLS

A Newton step solves the normal equations

$$\left(\mathbf{X}^\top \mathbf{W} \mathbf{X}\right) \Delta\boldsymbol{\beta} \;=\; \mathbf{X}^\top(\mathbf{y} - \mathbf{p}), \qquad \boldsymbol{\beta}_{\text{new}} \;=\; \boldsymbol{\beta}_{\text{old}} + \Delta\boldsymbol{\beta}. \tag{5}$$

Equivalently, define the *working response*

$$\mathbf{z} \;=\; \boldsymbol{\eta} + \mathbf{W}^{-1}(\mathbf{y} - \mathbf{p}), \qquad \boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta},$$

and solve the weighted least-squares system (Iteratively Reweighted Least Squares, IRLS)

$$\boldsymbol{\beta}_{\text{new}} \;\in\; \arg\min_{\boldsymbol{\beta}} \, (\mathbf{z} - \mathbf{X}\boldsymbol{\beta})^\top \mathbf{W} (\mathbf{z} - \mathbf{X}\boldsymbol{\beta}) \;\Longleftrightarrow\; \left(\mathbf{X}^\top \mathbf{W} \mathbf{X}\right) \boldsymbol{\beta}_{\text{new}} = \mathbf{X}^\top \mathbf{W} \mathbf{z}. \tag{6}$$

1.7 Regularization and MAP Interpretation

Add ℓ_2 (ridge) regularization (optionally excluding the intercept) with $\lambda \geq 0$:

$$J_\lambda(\boldsymbol{\beta}) \;=\; J(\boldsymbol{\beta}) + \frac{\lambda}{2}\,\|\tilde{\mathbf{R}}\boldsymbol{\beta}\|_2^2, \qquad \tilde{\mathbf{R}} = \mathrm{diag}(0, 1, \ldots, 1). \tag{7}$$

Then

$$\nabla J_\lambda = \mathbf{X}^\top(\mathbf{p} - \mathbf{y}) + \lambda\,\tilde{\mathbf{R}}^\top\tilde{\mathbf{R}}\boldsymbol{\beta}, \qquad \nabla^2 J_\lambda = \mathbf{X}^\top\mathbf{W}\mathbf{X} + \lambda\,\tilde{\mathbf{R}}^\top\tilde{\mathbf{R}},$$

and the IRLS system becomes

$$\left(\mathbf{X}^\top\mathbf{W}\mathbf{X} + \lambda\,\tilde{\mathbf{R}}^\top\tilde{\mathbf{R}}\right)\boldsymbol{\beta}_{\text{new}} \;=\; \mathbf{X}^\top\mathbf{W}\mathbf{z}. \tag{8}$$

MAP view: (7) equals $-\log p(\boldsymbol{\beta} \mid \text{data})$ for a Gaussian prior on $\boldsymbol{\beta}$ (zero-mean, precision proportional to $\tilde{\mathbf{R}}^\top\tilde{\mathbf{R}}$).

1.8 Numerical Stability: Log-Sum-Exp

Computing $\log\sum_i e^{x_i}$ appears frequently in machine learning, particularly in the **softmax** and **cross-entropy** functions. However, direct computation can be numerically unstable when x_i contains large values, leading to overflow. Let $C = \max_i x_i$; then

$$\begin{aligned}
\log\sum_i e^{x_i - C} &= \log\sum_i \frac{e^{x_i}}{e^C} \\
&= \log\frac{\sum_i e^{x_i}}{e^C} \\
&= \log[\sum_i e^{x_i}] - \log e^C \\
\log\sum_i e^{x_i - C} &= \log[\sum_i e^{x_i}] - C \\
\log[\sum_i e^{x_i}] &= \log[\sum_i e^{x_i - C}] + C
\end{aligned} \tag{9}$$

By subtracting C, the largest exponent becomes $e^0 = 1$, preventing overflow while preserving the correct value. This *Log-Sum-Exp* trick stabilizes the softmax and cross-entropy computations below.

Softmax. For logits $\{x_i\}$,

$$\mathrm{softmax}(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}, \quad \sum_j e^{x_j} \text{ computed via Log-Sum-Exp.}$$

Cross-Entropy with Softmax. For targets $\{p_i\}$ and logits $\{x_i\}$,

$$\begin{aligned}
L(p, \mathrm{softmax}(x)) &= -\sum_i p_i \log\frac{e^{x_i}}{\sum_j e^{x_j}} \\
&= -\sum_i p_i \big[x_i - \log\sum_j e^{x_j}\big] \\
&= -\sum_i p_i \big[x_i - \mathrm{LSE}(x)\big],
\end{aligned} \tag{10}$$

where $\mathrm{LSE}(x) = \log\sum_j e^{x_j}$ is implemented stably. *Practical note:* common losses (`CrossEntropyLoss`, `BCEWithLogitsLoss`) already combine softmax/ sigmoid with Log-Sum-Exp; do not add an explicit softmax/sigmoid layer before them.

1.9 Decision Rule, Calibration, and Separation

The probabilistic predictor is $\hat{p}(\mathbf{x}) = \sigma(\tilde{\mathbf{x}}^\top \widehat{\boldsymbol{\beta}})$. With equal misclassification costs, predict $\widehat{y} = \mathbf{1}\{\hat{p}(\mathbf{x}) \geq 1/2\}$ (decision boundary $\tilde{\mathbf{x}}^\top \widehat{\boldsymbol{\beta}} = 0$). For asymmetric costs/prior π, the optimal threshold is $t = \frac{c_{01}\pi}{c_{01}\pi + c_{10}(1-\pi)}$. If the data are strictly linearly separable, the MLE is not finite (coefficients diverge along a separating direction); ridge regularization (or early stopping) yields a well-posed solution.

1.10 Class/ Sample Weights

Class or sample weights $s_i \geq 0$ modify (2) to $\sum_i s_i\big(-y_i \log p_i - (1-y_i)\log(1-p_i)\big)$, which yields gradient $\mathbf{X}^\top \mathbf{S}(\mathbf{p} - \mathbf{y})$ and Hessian $\mathbf{X}^\top(\mathbf{S}\mathbf{W})\mathbf{X}$ with $\mathbf{S} = \mathrm{diag}(s_i)$.

1.11 Multiclass (Softmax) Logistic Regression

For K classes, define logits $\eta_{ik} = \tilde{\mathbf{x}}_i^\top \boldsymbol{\beta}_k$ and probabilities

$$p_{ik} = \frac{e^{\eta_{ik}}}{\sum_{t=1}^K e^{\eta_{it}}}, \qquad \sum_{k=1}^K p_{ik} = 1,$$

with one-hot targets y_{ik}. The cross-entropy is

$$J(\{\boldsymbol{\beta}_k\}) = \sum_{i=1}^n \Big(-\sum_{k=1}^K y_{ik}\eta_{ik} + \log \sum_{t=1}^K e^{\eta_{it}} \Big),$$

the gradient is $\nabla_{\boldsymbol{\beta}_k} J = \mathbf{X}^\top(\mathbf{p}_{\cdot k} - \mathbf{y}_{\cdot k})$, and the Hessian has blocks $\mathbf{X}^\top \mathbf{W}_{k\ell}\mathbf{X}$ with $\mathbf{W}_{k\ell} = \mathrm{diag}\big(p_{ik}(\mathbf{1}\{k = \ell\} - p_{i\ell})\big)$. Ridge adds $\lambda \sum_k \|\tilde{\mathbf{R}}\boldsymbol{\beta}_k\|_2^2$.

1.12 Algorithm (Newton / IRLS)

1. Initialize $\boldsymbol{\beta}^{(0)}$.
2. For $t = 0, 1, 2, \ldots$ until convergence:
   (a) $\boldsymbol{\eta}^{(t)} = \mathbf{X}\boldsymbol{\beta}^{(t)}$, $\mathbf{p}^{(t)} = \sigma(\boldsymbol{\eta}^{(t)})$, $\mathbf{W}^{(t)} = \mathrm{diag}(p_i^{(t)}(1 - p_i^{(t)}))$.
   (b) $\mathbf{z}^{(t)} = \boldsymbol{\eta}^{(t)} + (\mathbf{W}^{(t)})^{-1}(\mathbf{y} - \mathbf{p}^{(t)})$.
   (c) Solve $\big(\mathbf{X}^\top \mathbf{W}^{(t)}\mathbf{X} + \lambda\, \tilde{\mathbf{R}}^\top \tilde{\mathbf{R}}\big)\, \boldsymbol{\beta}^{(t+1)} = \mathbf{X}^\top \mathbf{W}^{(t)} \mathbf{z}^{(t)}$.
3. Output $\widehat{\boldsymbol{\beta}} = \boldsymbol{\beta}^{(t+1)}$.

1.13 Summary of Variables and Dimensions

- $\mathbf{x}_i \in \mathbb{R}^d$ (feature), $y_i \in \{0, 1\}$ (label), n samples, d features.
- $\mathbf{w} \in \mathbb{R}^d$, $b \in \mathbb{R}$; augmented $\tilde{\mathbf{x}}_i \in \mathbb{R}^{d+1}$, $\boldsymbol{\beta} \in \mathbb{R}^{d+1}$.
- $\mathbf{X} \in \mathbb{R}^{n \times (d+1)}$ stacks $\tilde{\mathbf{x}}_i^\top$; $\mathbf{y}, \mathbf{p} \in \mathbb{R}^n$.
- $\mathbf{W} \in \mathbb{R}^{n \times n}$ diagonal with entries $p_i(1 - p_i)$; $\tilde{\mathbf{R}} = \mathrm{diag}(0, 1, \ldots, 1)$.
- $J(\cdot)$: scalar cross-entropy; $\nabla J \in \mathbb{R}^{d+1}$; $\nabla^2 J \in \mathbb{R}^{(d+1)\times(d+1)}$.

1.14 Summary

Starting from a Bernoulli likelihood and the logistic function, we obtained the convex cross-entropy objective Eqn. (2), its gradient Eqn. (3) and Hessian Eqn. (4), and derived Newton/IRLS updates in Eqn. (6). Ridge regularization admits a MAP interpretation and stabilizes estimation under separation Eqn. (8). The Log-Sum-Exp trick ensures numerical stability Eqn. (9), and the softmax generalizes logistic regression to multiclass. Use losses that fuse normalization with cross-entropy to avoid redundant softmax/sigmoid layers and numerical issues Eqn. (10).