Chapter 1 Logistic Regression

Introduction

- *Binary Classification Problem*
- *Cross Entropy*
- *Sigmoid Regression*
- *Maximum a posteriori*

1.1 Classification

Example 1.1 Binary Classification Problem

Settings.

- Dataset: $D = \{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^d$ and $y_i \in \{0, 1\}$. Here, y_i denotes the classification target, while x_i represents the input features used to predict y_i.
- Model in Logistic Regression: In logistic regression, we start with a linear model $f(x) = w^\top x + b$. Unlike in ordinary regression, where the target variable lies in $\mathbb{R}$, here the target space collapses to $\{0, 1\}$. Thus, we need a function that maps real-valued outputs into this discrete set. Moreover, in many applications it is desirable to obtain not only a hard classification decision (0 or 1), but also a *soft* prediction: the probability of each class. Such a probabilistic interpretation provides both the likelihood estimate and the corresponding classification outcome.

Can we directly use a linear model to fit $p(y = 1 \mid x = x_i)$, as we did in the previous chapter? The answer is *no*. This is because there is a mismatch between the range of a linear model output (which lies in $\mathbb{R}$) and the valid domain of probabilities, $[0, 1]$.

To resolve this issue, we introduce a transformation function called the **sigmoid** function. The sigmoid maps any real-valued input into the interval $[0, 1]$, making it suitable for modeling probabilities. It is defined as:

Definition 1.1 (Sigmoid Function)

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{1.1}$$

♣

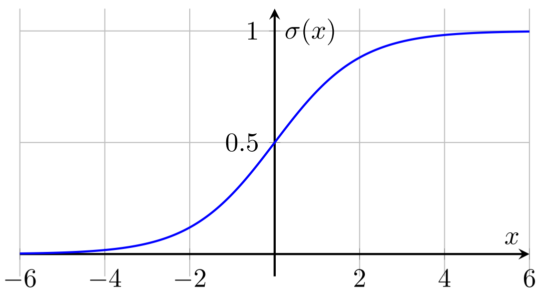

Figure 1.1: The sigmoid function $\sigma(z)$ over the interval $z \in [-6, 6]$.

The sigmoid function enjoys several elegant properties.

Theorem 1.1

$$1 - \sigma(z) = \sigma(-z)$$

♡

Proof This follows directly from the definition of $\sigma(z)$, or equivalently, by observing the symmetry of its graph.

To convert soft prediction results into binary outputs $\{0, 1\}$, we introduce a threshold: when $\sigma(z) = 0.5$, the model makes a hard prediction.

Definition 1.2 (Separating Hyperplane)

The condition $\sigma(z) = 0.5$ defines the separating hyperplane. *It partitions the input space $\mathbb{R}^d$ into two regions, thereby transforming probabilistic predictions into binary classification outcomes.* ♣

The normal vector w is perpendicular to this hyperplane and points towards the region where the model predicts class 1 (i.e., where $p(y = 1 \mid x) > 0.5$). We ensure this property by choosing the orientation of w accordingly.

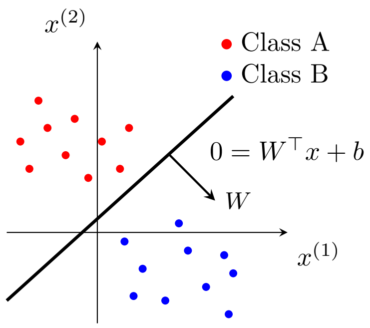

Figure 1.2: Classification by heperplane.

Model definition is clear, and now we turn our attention to parameter optimization. The question is: how can we find the proper w, b that achieve the best performance? The key problem here is to identify a suitable loss function that can be optimized via gradient descent.

Notice that:

$$P(y = 1 \mid x = x_i) = \sigma(f(x_i)) = \frac{1}{1 + \exp(-w^\top x + b)}. \tag{1.2}$$

We introduce a new method rather than continuing with ERM, by using **Maximum Likelihood Estimation (MLE)**. MLE is naturally designed to address probability modeling problems.

Definition 1.3 (Maximum Likelihood Estimation (MLE))

MLE aims to find parameters such that the likelihood of $P(y = y_i \mid x = x_i)$ is maximized. ♣

Remark For brevity, we write $P(y = y_i \mid x = x_i)$ as $P(y_i \mid x_i)$.

Definition 1.4 (Likelihood)

The likelihood on the entire training data is defined as

$$\prod_{i \in [n]} P(y_i \mid x_i; w, b) \tag{1.3}$$

assuming the samples are independent. ♣

According to the discussion above, in logistic regression we :

$$P(y_i \mid x_i) = \begin{cases} \sigma(w^\top x_i + b), & y_i = 1, \\ 1 - \sigma(w^\top x_i + b), & y_i = 0. \end{cases} \tag{1.4}$$

Therefore, the likelihood function can be expanded as:

$$\prod_{i \in [n]} \sigma(w^\top x_i + b)^{y_i} \left(1 - \sigma(w^\top x_i + b)\right)^{1 - y_i} \tag{1.5}$$

The above form is intuitive once we recall that $x^0 = 1$.

To ensure floating-point precision, given the large amount of data and the monotonicity of the logarithm function, we transform the likelihood into the maximization of the log-likelihood:

$$\operatorname*{argmax}_{w,b} \sum_{i \in [n]} \left[y_i \log \sigma(w^\top x_i + b) + (1 - y_i) \log \left(1 - \sigma(w^\top x_i + b)\right) \right] \tag{1.6}$$

This is the final objective in MLE.

MLE can be transformed into ERM by applying argmin to the negative log-likelihood. We define the **Cross-entropy Loss**:

$$\mathcal{L}(w,b) := -\sum_{i\in[n]}\left[y_i \log\sigma(w^\top x_i + b) + (1-y_i)\log\left(1-\sigma(w^\top x_i + b)\right)\right] \tag{1.7}$$

Thus, minimizing the cross-entropy loss is equivalent to maximizing the log-likelihood, making the equivalence between MLE and ERM immediate

Note *Why is the above loss called the Cross-entropy loss? The name originates from information theory. Entropy is defined as:*

$$H(P) = \sum_y P(y)\log\frac{1}{P(y)} = -\sum_y P(y)\log P(y). \tag{1.8}$$

In other words, rarer events (with smaller probability) carry more information, and entropy measures the expected amount of information. In our context, we can evaluate the information content of the prediction for $y=\hat{y}_i$ *given* $x = x_i$ *as:*

$$\begin{aligned} H(P) &= -\sum_{\hat{y}_i\in\{0,1\}} P(y=\hat{y}_i \mid x_i)\,\log P(y=\hat{y}_i \mid x_i) \\ &= -[P(y=1\mid x_i)\,\log P(y=1\mid x_i) + P(y=0\mid x_i)\,\log P(y=0\mid x_i)] \end{aligned} \tag{1.9}$$

Notice that under our estimation, $P(y=1\mid x_i) = \sigma(f(x_i;w,b))$ *and* $P(y=0\mid x_i) = 1-\sigma(f(x_i;w,b))$. *In practice, we substitute the empirical distribution of samples for the true distribution when comparing the negative log-likelihood with entropy. This is why the terminology of entropy from information theory is carried over to name this loss term.*

Note *The formal definition of cross entropy between two propbability distributions* q *and* p $H(q,p)$ *is defined by:*

$$H(q,p) = -\sum_y q(y)\log p(y) \tag{1.10}$$

Definition 1.5 (KL-Divergence)

The KL-Divergence between two distributions p *and* q *is defined by:*

$$\mathrm{KL}(q\|p) = \sum_y q(y)\log\frac{q(y)}{p(y)} \tag{1.11}$$

♣

KL-Divergence measures the difference between two given distributions. In particular, $\mathrm{KL}(q\|p)$ differs from the cross-entropy by only a constant term $H(q)$:

$$\mathrm{KL}(q\|p) = H(q,p) - H(q). \tag{1.12}$$

In other words, KL-Divergence quantifies the extra number of bits required when we use p to approximate the ground-truth distribution q.

Back to the main content. Recall that a closed-form solution can be derived for the linear regression problem, as mentioned in the previous chapter. Here, we would like to investigate whether logistic regression also admits a closed-form solution.

Following the same steps we applied in linear case, we define $\hat{x} = (x^\top, 1)^\top \in \mathbb{R}^{d+1}$ and $\hat{w} = (w^\top, b)^\top \in \mathbb{R}^{d+1}$, thus $f(x) = \hat{w}^\top\hat{x}$:

$$\begin{aligned} \mathcal{L}(\hat{w}) &= -\sum_{i\in[n]} y_i\log\sigma(\hat{w}\hat{x}_i) + (1-y_i)\log(1-\sigma(\hat{w}^\top\hat{x}_i)) \\ &= -\sum_{i\in[n]} y_i\log\frac{1+\exp(\hat{w}^\top\hat{x}_i)}{1+\exp(-\hat{w}^\top\hat{x}_i)} - \log(1+\exp(\hat{w}^\top\hat{x}_i)) \\ &= -\sum_{i\in[n]} y_i(\hat{w}^\top\hat{x}_i) - \log(1+\exp(\hat{w}^\top\hat{x}_i)) \end{aligned} \tag{1.13}$$

Take the derivative:

$$\frac{\partial \mathcal{L}(\hat{w})}{\partial \hat{w}} = -\sum_{i\in[n]} \left[y_i \hat{x}_i - \frac{\exp(\hat{w}^\top \hat{x}_i)}{1+\exp(\hat{w}^\top \hat{x}_i)} \hat{x}_i \right] \tag{1.14}$$

$$= -\sum_{i\in[n]} \left[y_i - P(y=1 \mid x_i) \right] \hat{x}_i \tag{1.15}$$

By gradient descent, the parameter is updated as $\hat{w} \leftarrow \hat{w} + \alpha \sum_{i\in[n]} \left(y_i - P(y_i = 1 \mid x_i) \right) \hat{x}_i$. This makes sense, since the term $y_i - P(y_i = 1 \mid x_i)$ directly measures the prediction error on sample i, and the update moves $\hat{w}$ a small step along the direction of the input $\hat{x}_i$ to reduce this error.

If for all i, we have $y_i = P(y = 1 \mid x_i)$, then the model predicts every label y_i perfectly. At this point, optimization reaches a stationary solution. If the training data admits such a perfect solution:

Definition 1.6 (linearly separable)

If all points can be separated by a linear model without error, we say the dataset is linearly separable. ♣

Example 1.3 is linearly separable, and the final state leads to $\|W\| \to \infty, \quad \|b\| \to \infty$. However, this situation is not desirable in practice, since it implies poor robustness. Hence a natural question arises: under the condition of linear separability, how can we find a well-chosen separating hyperplane that maximizes robustness? The answer will be presented in the next chapter, where we introduce the Support Vector Machine (SVM). The SVM optimizes $\hat{w}, \hat{b}$ by maximizing the margin (the distance between data points and the separating hyperplane), instead of simply minimizing the cross-entropy loss.

Although logistic regression may suffer from divergence of parameters under separable data, it often achieves better performance than SVM in practice, due to the following reasons:

1. In most real-world problems, the data are not linearly separable;
2. Applying L_2 regularization can effectively prevent parameter divergence.

Remark Why can't we use squared loss for classification? The reason is that in classification tasks such as logistic regression, the label $y \in \{0, 1\}$ should be interpreted as a categorical outcome rather than a numerical quantity.

Example 1.2 Multi-Class Classificaiton (Softmax Regression)

We can combine k sub-classifiers to solve a k-class classification task. Specifically, we apply a sigmoid-like transformation to each sub-linear model $f_k(x) = w_k^\top x + b_k$, and obtain the probability of class k using a normalized expression:

$$P(y = k \mid x) = \frac{\exp(w_k^\top x + b_k)}{\sum_{j=1}^{k} \exp(w_j^\top x + b_j)}. \tag{1.16}$$

Note *There is a close analogy between* ***softmax regression*** *in machine learning and the* ***partition function*** *in statistical physics.*

In softmax regression, the probability of assigning an input x to class k is

$$P(y = k \mid x) = \frac{\exp(\theta_k^\top x)}{\sum_{j=1}^{K} \exp(\theta_j^\top x)}. \tag{1.17}$$

Here the denominator

$$Z(x) = \sum_{j=1}^{K} \exp(\theta_j^\top x) \tag{1.18}$$

serves as a normalizing constant*, ensuring that the probabilities over all classes sum to* 1.

In statistical physics, for a system with possible states s of energy $E(s)$, the probability of observing state s under the Boltzmann distribution is

$$P(s) = \frac{\exp(-\beta E(s))}{Z} = -\frac{1}{\beta}\frac{\partial \log Z}{\partial E(s)}, \quad Z = \sum_s \exp(-\beta E(s)), \tag{1.19}$$

where Z is the partition function. *It normalizes the distribution and encodes all thermodynamic properties of the system.*

Thus, the role of $Z(x)$ in softmax regression is mathematically analogous to the role of the partition function Z in *statistical physics: both are log-sum-exp normalizers that transform unnormalized scores (energies or logits) into proper probability distributions.*

The softmax regression method has several advantages:

1. It is unified and normalized: $\sum_{j\in[j]} P(y=k \mid x) = 1$;
2. The amplify effect of exp function: when $f_k(x) \gg f_j(x), \forall j \neq k$, then $P(y=k \mid x) = 1$.

We apply the same MLE procedure to the multi-class classification task, which leads to the following optimization problem:

$$\underset{\{w_k, b_k\}}{\operatorname{argmax}} \sum_{i\in[n]} \log \frac{\exp(w_k^\top x_i + b_k)}{\sum_{j=1}^{k} \exp(w_j^\top x_i + b_j)} \tag{1.20}$$

Note *In modern neural networks, softmax regression is widely used as the standard classification method.*

Remark When $k = 2$, softmax regression reduces to logistic regression via the reparameterization $w = w_1 - w_2$ and $b = b_1 - b_2$.

1.2 Rethink of Linear Regression

Example 1.3 MLE Explanation for Linear Regression

Definition 1.7 (Gaussian/Normal Distribution)

$$x \sim \mathcal{N}(\mu, \sigma^2) \quad \Leftrightarrow \quad P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{1.21}$$

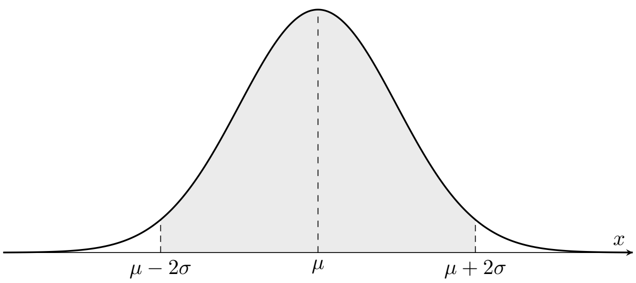

Figure 1.3: Normal distribution (95%).

While the Central Limit Theorem (CLT) does not imply that most datasets are normally distributed, it motivates modeling additive noise as Gaussian. We assume:

$$y = \underbrace{w^\top x + b}_{\text{latent model}} + \underbrace{\varepsilon}_{\text{noise}}, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2) \tag{1.22}$$

where σ^2 is a hyperparameter characterizing the noise scale. Then:

$$P(y \mid x; w, b, \sigma^2) = \mathcal{N}\big(y \,\big|\, w^\top x + b,\ \sigma^2\big) \tag{1.23}$$

The log-likelihood takes the form:

$$\begin{aligned} \underset{w,b}{\operatorname{argmax}} \sum_{i\in[n]} \log \mathcal{N}\big(y_i \,\big|\, w^\top x_i + b,\ \sigma^2\big) &= \underset{w,b}{\operatorname{argmax}} \sum_{i\in[n]} \left[-\tfrac{1}{2}\log(2\pi\sigma^2) - \frac{\big(y_i - (w^\top x_i + b)\big)^2}{2\sigma^2}\right] \\ &\Leftrightarrow \underset{w,b}{\operatorname{argmin}} \sum_{i\in[n]} \big(y_i - (w^\top x_i + b)\big)^2, \end{aligned} \tag{1.24}$$

where the equivalence follows by dropping constants and positive scalings. Equation (1.24) recovers ERM with the squared-loss objective.

Example 1.4 Maximum a Posteriori (MAP)

In the MLE perspective, w and b are treated as unknown fixed constants. In the Bayesian framework, however, even w, b are considered as random variables (R.V.). A "fixed constant" can be seen as a random variable with an extremely sharp distribution (near δ-distribution).

Suppose we place a Gaussian prior $P(\hat{w}) = \mathcal{N}(\hat{w} \mid 0,\ \sigma_w^2 \mathbb{I})$. The likelihood is given by

$$P(y \mid x, \hat{w};\, \sigma^2, \sigma_w^2) = \mathcal{N}\big(y \mid \hat{w}^\top \hat{x},\ \sigma^2\big) \tag{1.25}$$

We want to compute the posterior distribution of $\hat{w}$. By Bayes' rule:

$$\begin{aligned} P(\hat{w} \mid y, x) &= \frac{P(y \mid x, \hat{w})\, P(\hat{w})}{P(y \mid x)} \\ &\propto \left(\prod_{i \in [n]} P(y_i \mid x_i, \hat{w}) \right) P(\hat{w}) \end{aligned}$$

Expanding this expression:

$$P(\hat{w} \mid y, x) = \frac{1}{Z} \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right)^n \exp\!\left(-\frac{1}{2\sigma^2} \sum_{i \in [n]} \big(y_i - \hat{w}^\top \hat{x}_i\big)^2 \right) \left(\frac{1}{\sqrt{2\pi\sigma_w^2}} \right)^{d+1} \exp\!\left(-\frac{\hat{w}^\top \hat{w}}{2\sigma_w^2} \right)$$

Taking the negative logarithm of the posterior, note that only the terms involving $\hat{w}$ are subject to optimization:

$$-\log P(\hat{w} \mid y, x) = \sum_{i \in [n]} \big(y_i - \hat{w}^\top \hat{x}_i\big)^2 + \frac{\sigma^2}{\sigma_w^2} \|\hat{w}\|^2 + \text{Const.}$$

Letting $\lambda = \sigma^2 / \sigma_w^2$, the MAP estimator is obtained by:

$$\operatorname*{argmin}_{\hat{w}} \sum_{i \in [n]} \big(y_i - \hat{w}^\top \hat{x}_i\big)^2 + \lambda \|\hat{w}\|^2, \tag{1.26}$$

which is exactly ridge regression. Therefore, the internal consistency of the theory is demonstrated.