

Chapter 1 Logistic Regression

Introduction



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)

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

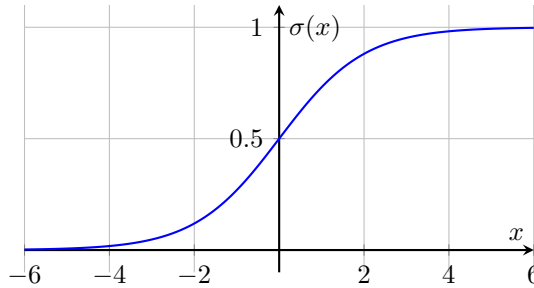


Figure 1.1: The sigmoid function $\sigma(z)$ over the interval $z \in [-6, 6]$.

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 $p(y = 1 | x) > 0.5$). We ensure this property by choosing the orientation of w accordingly.

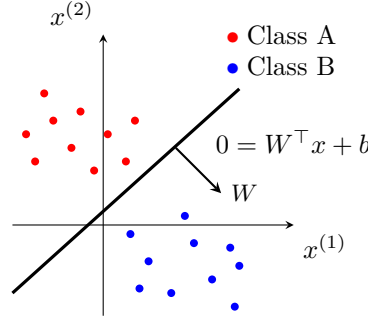


Figure 1.2: Classification by heperplane.

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:

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

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 $P(y = y_i | x = x_i)$ is maximized.

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

Definition 1.4 (Likelihood)

The likelihood on the entire training data is defined as

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

assuming the samples are independent.

According to the discussion above, in logistic regression we have

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

Therefore, the likelihood function can be expanded as

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

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:


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

This is the final objective in MLE.

MLE can be transformed into ERM by applying argmin to the negative log-likelihood. Thus we define the **Cross-**

entropy Loss:

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


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

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

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:

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

Notice that $P(y = 1 | x_i) = \sigma(f(x_i; w, b))$ and $P(y = 0 | x_i) = 1 - \sigma(f(x_i; w, b))$. 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 probability distributions q and p $H(q, p)$ is defined by:

$$H(q, p) = - \sum_y q(y) \log p(y) \quad (1.10)$$

Definition 1.5 (KL-Divergence)

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

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

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

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

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. Rewind that closed-form solution can be extract from linear regression problem as we mentioned in last chapter, we hope to find out whether logistic regression has 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 $f(x) = \hat{w}^\top \hat{x}$:

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

Derivate:

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

$$= - \sum_{i \in [n]} [y_i - P(y = 1 | x_i)] \hat{x}_i \quad (1.15)$$

By gradient descent, the parameter is updated as $\hat{w} \leftarrow \hat{w} + \alpha \sum_{i \in [n]} (y_i - P(y_i = 1 | x_i)) \hat{x}_i$. 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, that is, if all points can be separated by a linear model without error, we say the dataset is **linearly separable**.

Example 1.2 is linearly separable, and the final state leads to $\|W\| \rightarrow \infty$, $\|b\| \rightarrow \infty$. 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.