Machine Learning

Author: Shaoheng Yan (PhotonYan)

Institute: Institute of Artificial Intelligence, Peking University

Date: 2025-2026 Autumn

Contents

Chapter 1 Linear Regression

Introduction

- ❑ *ERM*
- ❑ *Gradient Descent*
- ❑ *Ridge Regression (L_2 regularization)*
- ❑ *Lasso Regression (L_1 regularization)*

1.1 Basic Knowledge

Example 1.1 Linear Regression

Settings.

- Dataset: $D = \{(x_i, y_i)\}_{i=1}^{n}$, where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. Here, y_i denotes the regression target, while x_i represents the input features used to predict y_i.
- Linear Model: $f(x) = W^\top x + b$, with weight $W \in \mathbb{R}^d$ and bias $b \in \mathbb{R}$.

Note *This definition is equivalent to an inner product: $\hat{y} = W^\top x + b$.*

Definition 1.1 (Learnable / Trainable Parameters)

Learnable parameters are those that can be updated during the training process.

Quiz. How to determine whether a parameter is learnable? Quiz: How to determine W and b?

Ans: **ERM** (Empirical Risk Minimization)

- Loss function. Squared Loss (SE) is commonly used during optimization. The training objective can be written as:

$$\operatorname*{argmin}_{W,b} \frac{1}{n} \sum_{i\in[n]} \left(y_i - (W^\top x_i + b)\right)^2 \tag{1.1}$$

The blue factor $1/n$ can be omitted in theoretical analysis, but is often kept in practice to stabilize the loss function during implementation.

Quiz: How to optimize the parameters?

Ans: **Gradient Descent** (as a traditional ML method). In the case of linear regression:

$$\frac{\partial \mathcal{L}}{\partial b} = -2 \sum_{i\in[n]} (y_i - W^\top x_i - b) \tag{1.2}$$

$$\frac{\partial \mathcal{L}}{\partial W} = -2 \sum_{i\in[n]} (y_i - W^\top x_i - b)x_i \tag{1.3}$$

Note *In the field of machine learning, the gradient of a scalar with respect to a vector is itself a vector* ***(not a covector)****. This means:*

$$\frac{\partial \mathcal{L}}{\partial W} = \begin{pmatrix} \frac{\partial \mathcal{L}}{\partial W_1} \\ \frac{\partial \mathcal{L}}{\partial W_2} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial W_d} \end{pmatrix} = \left(\frac{\partial \mathcal{L}}{\partial W_1}, \frac{\partial \mathcal{L}}{\partial W_2}, \cdots, \frac{\partial \mathcal{L}}{\partial W_d} \right)^\top \tag{1.4}$$

Note *Here are some commonly used derivative formulas:*

$$\frac{\partial x^\top x}{\partial x} = 2x \tag{1.5}$$

$$\frac{\partial a^\top x}{\partial x} = a, \quad \frac{\partial Ax}{\partial x} = A^\top \tag{1.6}$$

$$\frac{\partial x^\top Ax}{\partial x} = (A + A^\top)x \tag{1.7}$$

Remark Both sides of an equation must have the same dimension. This principle can be used as a consistency check.

We optimize the parameters by subtracting a scalar multiple of the gradient from the parameters, considering the physical meaning of the gradient: the direction of the steepest **increase**.

Definition 1.2 (Hyperparameter)

A parameter that is fixed during optimization and specified before the training process. ♣

That is:

$$W' = W - \alpha \frac{\partial \mathcal{L}}{\partial W}, \quad b' = b - \alpha \frac{\partial \mathcal{L}}{\partial b} \tag{1.8}$$

Optimization will stop when the norm of the parameter update becomes smaller than a given hyperparameter.

1.2 Closed-Form of Linear Regression

Proposition 1.1

*Linear Regression has **Closed-Form** solution.* ♡

Settings.

- Metrix $X_0 := (x_1^\top, \cdots, x_n^\top)^\top$;
- Metrix $X := (X_0, \mathbb{1}) \in \mathbb{R}^{n\times(d+1)}$;
- $y = (y_1, \cdots, y_n)^\top \in \mathbb{R}^n$;
- $\hat{w} = (w, b)^\top \in \mathbb{R}^{d+1}$.

Then the loss function of $\hat{w}$ can be writtern as:

$$\mathcal{L}(\hat{w}) = (y - X\hat{w})^\top (y - X\hat{w}) = \|y - X\hat{w}\|_2^2 \tag{1.9}$$

Here, $\|\cdot\|_p$ denotes the *p-norm* of a vector.

Note *Vectors can sometimes be treated as scalars, since linearity ensures that the validity of a proposition can be extended to any finite dimension.*

Notice that the optimization stops when $\partial\mathcal{L}(\hat{w})/\partial\hat{w} = 0$. Under this condition, the parameters can be solved from the above constraint by following steps:

$$\frac{\partial \mathcal{L}(\hat{w})}{\partial \hat{w}} = -2X^\top (y - X\hat{w}) \tag{1.10}$$

Note *Both dimensional analysis and calculation using Leibniz's rule lead to the same result as the formula above:*

$$\begin{aligned} \mathcal{L}(\hat{w}) &= y^\top y - 2y^\top X\hat{w} + \hat{w}^\top X^\top X\hat{w} \\ \partial_{\hat{w}} \mathcal{L}(\hat{w}) &= -2X^\top y + 2X^\top X\hat{w} \\ &= -2X^\top (y - X\hat{w}) \end{aligned}$$

Remark More matix formulas are available in Matrix Cookbook.

Thus, the target of the optimization satisfied:

$$X^\top y = X^\top X\hat{w} \tag{1.11}$$

That is:

$$\hat{w} = (X^\top X)^{-1} X^\top y \tag{1.12}$$

when $X^\top X$ invertible (non-singular / full-rank).

Example 1.2 When does $X^\top X$ not invertible?

Solution $X \in \mathbb{R}^{n\times(d+1)}$:

- $d + 1 > n$. *Brief Proof:* $\text{rank}\,(X^\top X) = \text{rank}\,(X) \le \min(n, d+1) = n < d + 1$.
- *X has repeated columns. Proof is trivial.*

When $X^\top X$ isn't invertible:

1. If rank $(X^\top X, X^\top y) >$ rank $(X^\top X)$, $\hat{w}$ has no solution;
2. $\hat{w}$ has infinity solution o.w.

Situation 1 is **Impossible** because both $X^\top X$ and $X^\top y$ can be represented in the column space of $X^\top$. Therefore, the optimization problem must have a solution, which may be either unique or infinite.

As an infinite set of solutions makes it difficult to determine which estimate of $\hat{w}$ to choose, we apply L_2 **regularization** to linear regression, which is commonly referred to as **Ridge Regression**. That is:

$$\mathcal{L}_{L_2} := \mathcal{L}(\hat{w}) + \boxed{\lambda \|\hat{w}\|_2^2} \tag{1.13}$$

Noticed that $\|\hat{w}\|_2^2 = \sum_{i=1}^{d+1} \hat{w}_i^2$, L_2 regularization prevents any single dimension from being assigned an excessively large weight, , and encourages the model to make use of more dimensions during training.

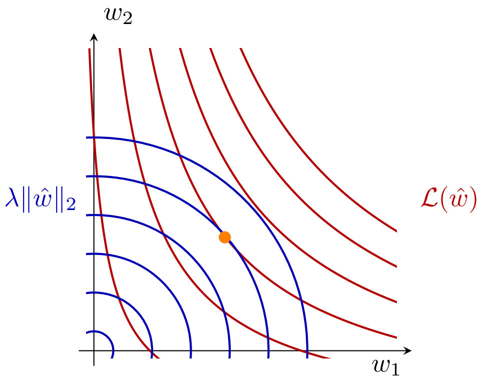

Figure 1.1: Illustration of L_2 regularization. The contours represent level sets of the regularized loss $\mathcal{L}(\hat{w}) + \lambda\|\hat{w}\|_2^2$, which take the form of concentric ellipses (circle in the plot).

During ridge regression, we minimize the $\mathcal{L}_{L_2}$:

$$\underset{\hat{w}}{\operatorname{argmin}}\, (y - X\hat{w})^\top (y - X\hat{w}) + \lambda \hat{W}^\top \hat{W} \tag{1.14}$$

The optimization stops when:

$$\frac{\partial \mathcal{L}_{L_2}}{\partial \hat{w}} = -2X^\top y + 2X^\top X\hat{w} + 2\lambda I\hat{w} = 0 \tag{1.15}$$

$$\Rightarrow (X^\top X + \lambda \mathbb{I})\hat{w} = X^\top w \tag{1.16}$$

Proposition 1.2

$X^\top X + \lambda \mathbb{I}$ always invertible.

Proof Since $X^\top X$ is a real symmetric matrix, we have the eigen-decomposition $X^\top X = U\Lambda U^\top$, where $\Lambda = \operatorname{diag}(\lambda_1, \cdots, \lambda_{d+1})$. Moreover, as $X^\top X \succeq 0$ is positive semi-definite, it follows that $\forall i \in [d+1]$, $\lambda_i \geq 0$. Note that:

$$\lambda \mathbb{I} = \lambda U U^\top \tag{1.17}$$

since U is an orthogonal matrix. Hence:

$$X^\top X + \lambda \mathbb{I} = U(\Lambda + \lambda I)U^\top \tag{1.18}$$

For all $i \in [d+1]$, we have:

$$\lambda_i + \lambda > \lambda_i \geq 0 \tag{1.19}$$

Thus, $X^\top X + \lambda I$ is a full-rank matrix.

Remark Numerical issues may still occur even if $X^\top X$ is full rank (e.g., when eigenvalues λ_k are close to zero). The L_2 regularization factor λ mitigates this issue by shifting the eigenvalues upward, thereby improving numerical stability during training.

Another regularization method often used is L_1 regularization, where the loss function is defined as:

$$\mathcal{L}_{L_1} := \mathcal{L}(\hat{w}) + \boxed{\lambda\|\hat{w}\|_1} \tag{1.20}$$

L_1 regularization can induce sparsity in $\hat{w}$, which works in contrast to L_2 regularization. Specifically, L_1 regularization encourages the model to rely on only a small subset of input features, effectively performing **feature selection**.

Linear regression with L_1 regularization is called **Lasso Regression** (Least Absolute Shrinkage and Selection Operator).

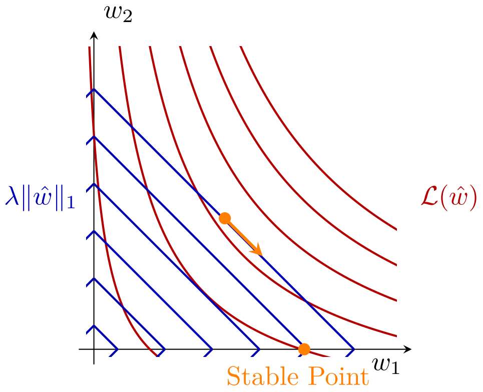

Figure 1.2: Illustration of L_1 regularization. The contours represent level sets of the regularized loss $\mathcal{L}(\hat{w}) + \lambda\|\hat{w}\|_1$, which take the form of nested diamonds (squares rotated by 45° in the plot).

1.3 Geomeric View of LR

Ideally, we would like to solve $X\hat{w} = y$. If y lies on the hypersurface

$$\mathcal{M}(X) := \text{Span}(X) = \{Xw : w \in \mathbb{R}^d\} \subset \mathbb{R}^n \tag{1.21}$$

then the equation admits an exact solution. In most cases, however, $y \notin \mathcal{M}(X)$, so no exact solution exists. Nevertheless, we can always find an estimator $\hat{w}$ such that $\mathcal{P}_{\mathcal{M}(X)}y = X\hat{w}$, where $\mathcal{P}_{\mathcal{M}(X)}$ denotes the orthogonal projection onto the hypersurface $\mathcal{M}(X)$.

Proposition 1.3

$$\hat{y} = X\hat{w} \quad \Rightarrow \quad \hat{w} \textit{ is solution to LR.} \tag{1.22}$$

♡

Proof

$$\begin{aligned} y - \hat{y} \perp \mathcal{M}(X) \quad &\Rightarrow \quad y - X\hat{w} \perp \mathcal{M}(X) \\ &\Rightarrow \quad X^\top(y - X\hat{w}) = 0 \quad \Rightarrow \quad \hat{w} = (X^\top X)^{-1}X^\top y \end{aligned}$$

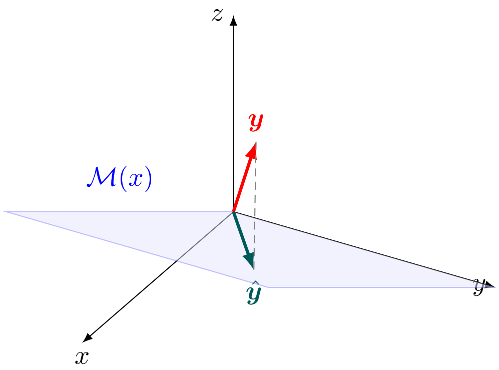

Figure 1.3: Orthogonal projection interpretation of linear regression. The predicted vector $X\hat{w}$ is obtained as the projection of y onto the hypersurface $\mathcal{M}(X) = \{Xw : w \in \mathbb{R}^d\}$, which is a linear subspace in the classical case.

Chapter 2 Logistic Regression

Introduction

❏

Example 2.1 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 2.1 (Sigmoid Function)

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \tag{2.1}$$

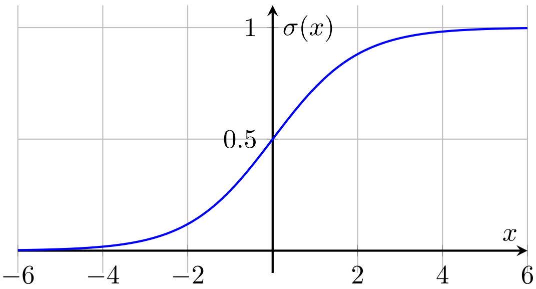

Figure 2.1: The sigmoid function $\sigma(z)$ over the interval $z \in [-6, 6]$.

The sigmoid function enjoys several elegant properties.

Theorem 2.1

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

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 2.2 (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.

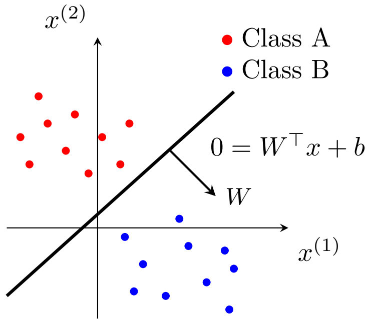

Figure 2.2: Classification by heperplane.

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:

$$P(y = 1 \mid x = x_i) = \sigma(f(x_i)) = \frac{1}{1 + \exp(-W^\top x + b)}. \tag{2.2}$$

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 2.3 (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 2.4 (Likelihood)

The likelihood on the entire training data is defined as

$$\prod_{i \in [n]} P(y_i \mid x_i; W, b), \tag{2.3}$$

assuming the samples are independent. ♣

According to the discussion above, in logistic regression we have

$$P(y_i \mid x_i) = \begin{cases} \sigma(W^\top x_i + b), & y_i = 1, \\ 1 - \sigma(W^\top x_i + b), & y_i = 0. \end{cases} \tag{2.4}$$

Therefore, the likelihood function can be expanded as

$$\prod_{i \in [n]} \sigma(W^\top x_i + b)^{y_i} \left(1 - \sigma(W^\top x_i + b)\right)^{1-y_i} \tag{2.5}$$

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:

$$\operatorname*{argmax}_{W,b} \sum_{i \in [n]} \left[y_i \log \sigma(W^\top x_i + b) + (1 - y_i) \log \left(1 - \sigma(W^\top x_i + b)\right) \right] \tag{2.6}$$

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:

$$\mathcal{L}(W,b) := -\sum_{i\in[n]} \left[y_i \log \sigma(W^\top x_i + b) + (1-y_i)\log\left(1-\sigma(W^\top x_i + b)\right)\right] \tag{2.7}$$

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

$$H(P) = \sum_y P(y)\log\frac{1}{P(y)} = -\sum_y P(y)\log P(y). \tag{2.8}$$

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:

$$\begin{aligned} H(P) &= -\sum_{\hat{y}_i\in\{0,1\}} P(y=\hat{y}_i \mid x_i)\,\log P(y=\hat{y}_i \mid x_i) \\ &= -[P(y=1\mid x_i)\,\log P(y=1\mid x_i) + P(y=0\mid x_i)\,\log P(y=0\mid x_i)] \end{aligned} \tag{2.9}$$

Notice that $P(y=1\mid x_i) = \sigma(f(x_i;W,b))$ 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:*

$$H(q,p) = -\sum_y q(y)\log p(y) \tag{2.10}$$

Definition 2.5 (KL-Divergence)

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

$$\mathrm{KL}(q\|p) = \sum_y q(y)\log\frac{q(y)}{p(y)} \tag{2.11}$$

♣

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 $H(q)$:

$$\mathrm{KL}(q\|p) = H(q,p) - H(q). \tag{2.12}$$

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.