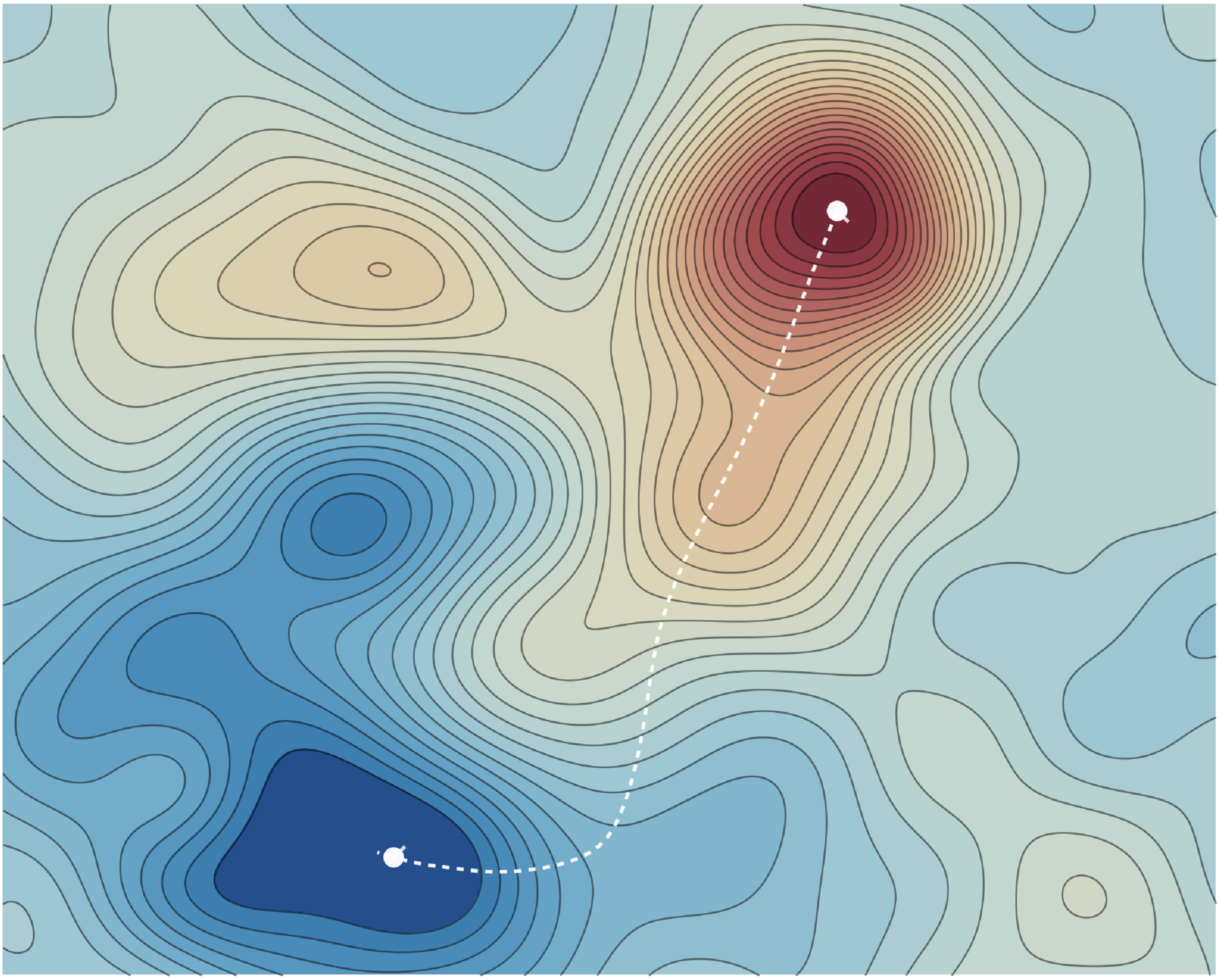

Machine Learning

Author: Shaoheng Yan (PhotonYan)

Instructor: Prof. Muhan Zhang

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:*

$$\frac{\partial \mathcal{L}}{\partial w} = \begin{pmatrix} \frac{\partial \mathcal{L}}{\partial w_1} \\ \frac{\partial \mathcal{L}}{\partial w_2} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial w_d} \end{pmatrix} = \left(\frac{\partial \mathcal{L}}{\partial w_1}, \frac{\partial \mathcal{L}}{\partial w_2}, \cdots, \frac{\partial \mathcal{L}}{\partial w_d} \right)^\top \tag{1.4}$$

See the definition of matix derivatives (assuming no special structure in the matrix) in Matrix Cookbook Chapter 2.

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.

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

Then the loss function of $\hat{w}$ can be written 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:* $\operatorname{rank}(X^\top X) = \operatorname{rank}(X) \le \min(n, d+1) = n < d+1$.
- X *has repeated columns. Proof is trivial.*

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

1. If $\operatorname{rank}(X^\top X, X^\top y) > \operatorname{rank}(X^\top X)$, $\hat{w}$ 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:

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

where $\lambda > 0$ is a hyperparameter. Notice 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.

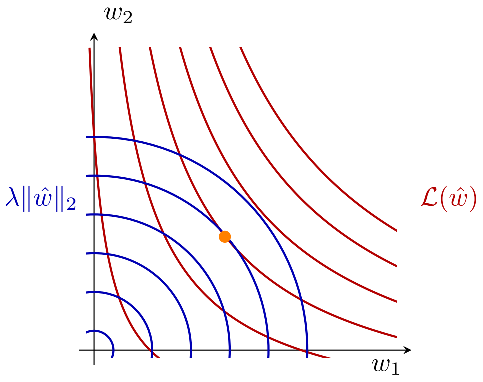

Figure 1.1: Illustration of L_2 regularization. The contours represent level sets of the regularized loss $\mathcal{L}(\hat{w}) + \lambda\|\hat{w}\|_2^2$, which take the form of concentric ellipses (circle in the plot).

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

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

The optimization stops when:

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

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

Proposition 1.2

$X^\top X + \lambda I$ 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 $X^\top X \succeq 0$ (positive semi-definite), it follows that $\forall i \in [d+1]$, $\lambda_i \ge 0$. Note that:

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

since U is an orthogonal matrix. Hence:

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

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).

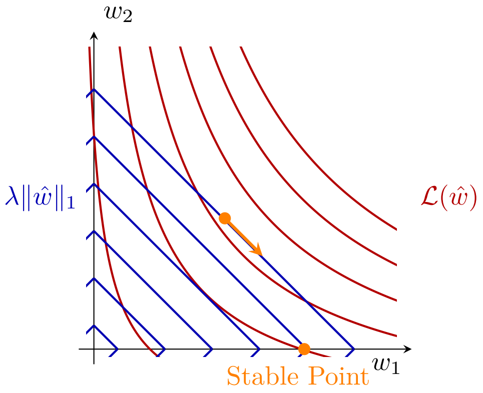

Figure 1.2: Illustration of L_1 regularization. The contours represent level sets of the regularized loss $\mathcal{L}(\hat{w}) + \lambda\|\hat{w}\|_1$, which take the form of nested diamonds (squares rotated by 45° in the plot).

1.3 Geomeric View of LR

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

$$\mathcal{M}(X) := \mathrm{Span}(X) = \{Xw : w \in \mathbb{R}^{d+1}\} \subset \mathbb{R}^n \tag{1.21}$$

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

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

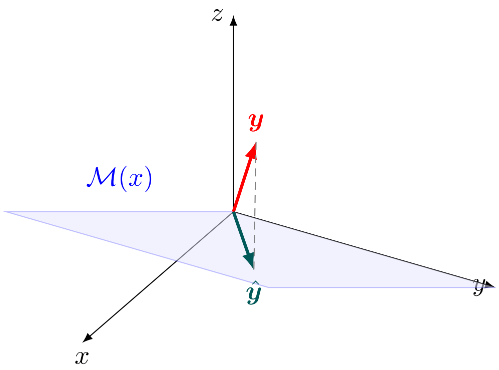

Figure 1.3: Orthogonal projection interpretation of linear regression. The predicted vector $X\hat{w}$ is obtained as the projection of y onto the hypersurface $\mathcal{M}(X) = \{Xw : w \in \mathbb{R}^{d+1}\}$, which is a linear subspace in the classical case.

Chapter 2 Logistic Regression

Introduction

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

2.1 Classification

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)

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

♣

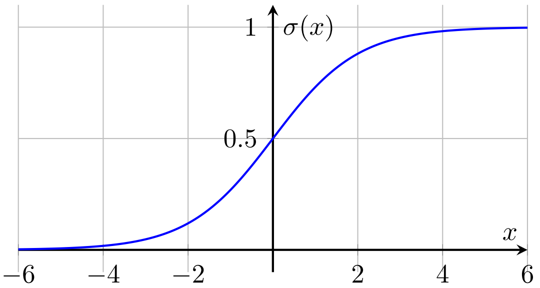

Figure 2.1: The sigmoid function $\sigma(z)$ over the interval $z \in [-6, 6]$.

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.

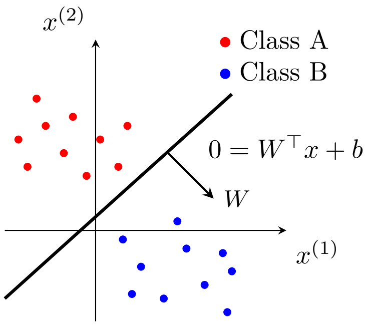

Figure 2.2: Classification by heperplane.

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 :

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

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:

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

This is the final objective in MLE.

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

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

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:*

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

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 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:*

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

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. 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 $f(x) = \hat{w}^\top \hat{x}$:

$$\begin{aligned} \mathcal{L}(\hat{w}) &= -\sum_{i\in[n]} y_i \log \sigma(\hat{w}\hat{x}_i) + (1 - y_i) \log(1 - \sigma(\hat{w}^\top \hat{x}_i)) \\ &= -\sum_{i\in[n]} y_i \log \frac{1 + \exp(\hat{w}^\top \hat{x}_i)}{1 + \exp(-\hat{w}^\top \hat{x}_i)} - \log(1 + \exp(\hat{w}^\top \hat{x}_i)) \\ &= -\sum_{i\in[n]} y_i(\hat{w}^\top \hat{x}_i) - \log(1 + \exp(\hat{w}^\top \hat{x}_i)) \end{aligned} \tag{2.13}$$

Take the derivative:

$$\frac{\partial \mathcal{L}(\hat{w})}{\partial \hat{w}} = -\sum_{i\in[n]} \left[y_i \hat{x}_i - \frac{\exp(\hat{w}^\top \hat{x}_i)}{1+\exp(\hat{w}^\top \hat{x}_i)} \hat{x}_i \right] \tag{2.14}$$

$$= -\sum_{i\in[n]} \left[y_i - P(y=1 \mid x_i) \right] \hat{x}_i \tag{2.15}$$

By gradient descent, the parameter is updated as $\hat{w} \leftarrow \hat{w} + \alpha \sum_{i\in[n]} \Big(y_i - P(y_i = 1 \mid x_i) \Big) \hat{x}_i$. 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 2.6 (linearly separable)

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

Example 3.3 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 2.2 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:

$$P(y = k \mid x) = \frac{\exp(w_k^\top x + b_k)}{\sum_{j=1}^{k} \exp(w_j^\top x + b_j)}. \tag{2.16}$$

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

$$P(y = k \mid x) \;=\; \frac{\exp(\theta_k^\top x)}{\sum_{j=1}^{K} \exp(\theta_j^\top x)}. \tag{2.17}$$

Here the denominator

$$Z(x) \;=\; \sum_{j=1}^{K} \exp(\theta_j^\top x) \tag{2.18}$$

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

$$P(s) \;=\; \frac{\exp(-\beta E(s))}{Z} \;=\; -\frac{1}{\beta}\frac{\partial \log Z}{\partial E(s)}, \quad Z \;=\; \sum_s \exp(-\beta E(s)), \tag{2.19}$$

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:

$$\underset{\{w_k, b_k\}}{\operatorname{argmax}} \sum_{i\in[n]} \log \frac{\exp(w_k^\top x_i + b_k)}{\sum_{j=1}^{k} \exp(w_j^\top x_i + b_j)} \tag{2.20}$$

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 $b = b_1 - b_2$.

2.2 Rethink of Linear Regression

Example 2.3 MLE Explanation for Linear Regression

Definition 2.7 (Gaussian/Normal Distribution)

$$x \sim \mathcal{N}(\mu, \sigma^2) \quad \Leftrightarrow \quad P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{2.21}$$

♣

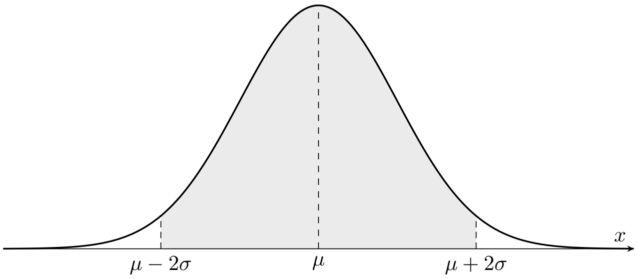

Figure 2.3: Normal distribution (95%).

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

$$y = \underbrace{w^\top x + b}_{\text{latent model}} + \underbrace{\varepsilon}_{\text{noise}}, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2) \tag{2.22}$$

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

$$P(y \mid x; w, b, \sigma^2) = \mathcal{N}\big(y \,\big|\, w^\top x + b,\ \sigma^2\big) \tag{2.23}$$

The log-likelihood takes the form:

$$\begin{aligned} \underset{w,b}{\operatorname{argmax}} \sum_{i\in[n]} \log \mathcal{N}\big(y_i \,\big|\, w^\top x_i + b,\ \sigma^2\big) &= \underset{w,b}{\operatorname{argmax}} \sum_{i\in[n]} \left[-\tfrac{1}{2}\log(2\pi\sigma^2) - \frac{(y_i - (w^\top x_i + b))^2}{2\sigma^2}\right] \\ &\Leftrightarrow \underset{w,b}{\operatorname{argmin}} \sum_{i\in[n]} \big(y_i - (w^\top x_i + b)\big)^2, \end{aligned} \tag{2.24}$$

where the equivalence follows by dropping constants and positive scalings. Equation (2.24) recovers ERM with the squared-loss objective.

Example 2.4 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

$$P(y \mid x, \hat{w};\, \sigma^2, \sigma_w^2) = \mathcal{N}\big(y \mid \hat{w}^\top \hat{x},\ \sigma^2\big) \tag{2.25}$$

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:

$$\operatorname*{argmin}_{\hat{w}} \sum_{i \in [n]} \big(y_i - \hat{w}^\top \hat{x}_i\big)^2 + \lambda \|\hat{w}\|^2, \tag{2.26}$$

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

Chapter 3 Support Vector Machine

Introduction

- ❑ *Conditioned optimization*
- ❑ *Inequality optimization*
- ❑ *KKT condition*
- ❑ *Support Vector Machine*

Support Vector Machine (SVM) is also a binary classification model, often introduced as an alternative to logistic regression, since—as we have discussed in the previous chapter—logistic regression has several notable limitations. SVM provides an elegant framework that allows us to transfer powerful results and intuition from optimization theory into machine learning.

3.1 Conditioned Optimization

Unlike the unconstrained optimization problems we studied earlier, the support vector machine is originally formulated as a **constrained optimization problem**. This distinction is fundamental. In mathematics, unconstrained optimization is defined as:

$$\underset{x}{\operatorname{argmin}} f(x) \tag{3.1}$$

When we deal with constrained optimization, we add an auxiliary constraint $h(x) = 0$ (where both $f(x)$ and $g(x)$ are differentiable). Considering the geometric meaning of the gradient, we observe:

1. For any point on the surface defined by $g(x)$, the gradient $\nabla g(x)$ must be orthogonal to the surface.
2. For any critical point x^* that achieves a local minimum, the gradient $\nabla f(x^*)$ must also be orthogonal to the surface.

Note *The second property can be understood by analogy with electrostatics: near a conductor, the electric field must be orthogonal to the surface. Otherwise, there would be a component of the field (gradient) along the constrained sub-hypersurface, violating equilibrium.*

Extending this analogy: in the electrostatic case, such a tangential component would induce currents along the conductor's surface. In the optimization setting, a tangential component corresponds to a gradient lying along the constraint manifold, implying that the objective function can still be further optimized without violating the constraint.

Moreover, the charge distribution problem in electrostatics can itself be interpreted as an energy minimization process: the charges rearrange to minimize the system's electrostatic energy. In this sense, the equilibrium state of charges on a conductor can be seen as an optimization problem, which strengthens the analogy with constrained optimization in mathematics.

Combining the two conditions above, we obtain the **necessary** condition for a local minimum point, as shown in Figure 3.1:

$$\exists \lambda \quad \text{s.t.} \quad \nabla f(x^*) + \lambda \nabla h(x^*) = 0 \tag{3.2}$$

Remark In some contour examples, the constraint condition does not cover the entire space. In such cases, the above equation may *not* always be satisfied.

Now we introduce the *Lagrangian function*, which plays a central role in optimization theory:

$$L(x, \lambda) := f(x) + \underbrace{\lambda}_{\text{Lagrange Multiplier}} h(x) \tag{3.3}$$

Extending to the case of multiple constraints, the Lagrangian is defined as:

$$L(x, \lambda) := f(x) + \sum_{i=1}^{k} \lambda_i h_i(x) \tag{3.4}$$

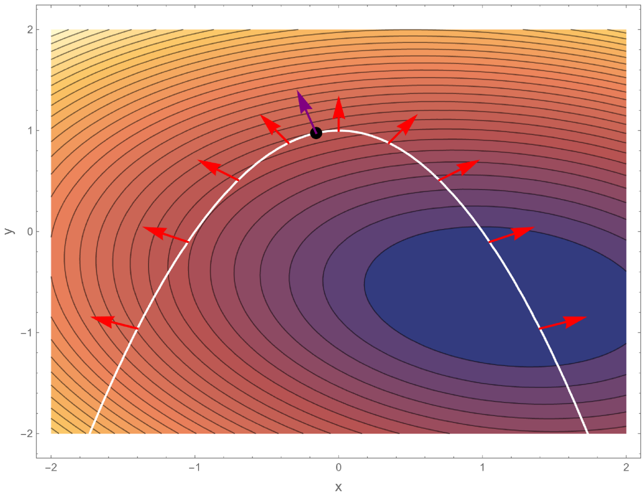

Figure 3.1: Illustration of the necessary condition for constrained local minima.

Theorem 3.1

If x^* is a local minimum point, then there exists a set of multipliers $\lambda = (\lambda_1, \ldots, \lambda_k)$ such that

$$\begin{cases} \nabla_x L(x^*, \lambda) = 0, \\ \nabla_\lambda L(x^*, \lambda) = 0. \end{cases} \tag{3.5}$$

The first condition corresponds to the stationarity (minimum) condition, while the second condition enforces the constraints.

Corollary 3.1

In multi-constrained problems, the minimum condition becomes

$$\nabla f(x^*) + \sum_{i=1}^{k} \lambda_i \nabla h_i(x^*) = 0. \tag{3.6}$$

Remark In higher dimensions, the constrained submanifold may possess certain free variables that can be chosen arbitrarily while still satisfying the minimum condition. Specifically, if $\nabla f(x^*)$ cannot be expressed as a linear combination of $\nabla h_1(x^*)$ and $\nabla h_2(x^*)$, it is possible to continue optimizing along the intersection submanifold defined by $h_1 = 0$ and $h_2 = 0$, as shown in Figure 3.2. Therefore, it must be emphasized that the above equation provides only a **necessary condition**, not a sufficient one.

Example 3.1 Consider the following constrained optimization problem:

$$\underset{x_1, x_2}{\operatorname{argmin}} \; x_1 + x_2 \tag{3.7}$$

subject to the constraint

$$(x_1 - 1)^2 + x_2^2 - 1 = 0 \tag{3.8}$$

Solution $1 - \sqrt{2}$.

Note *The Lagrange multiplier method can provide critical points, but these do not always correspond to the minimum solution.*

Note *In convex settings, any local minimum is also a global minimum.*

A new problem arises: what if we have an inequality constraint instead of a strict equality, i.e., $g(x) \leq 0$?

3.2 Inequality optimization

We assume the feasible set of x is compact; otherwise, the existence of a minimum may be problematic.

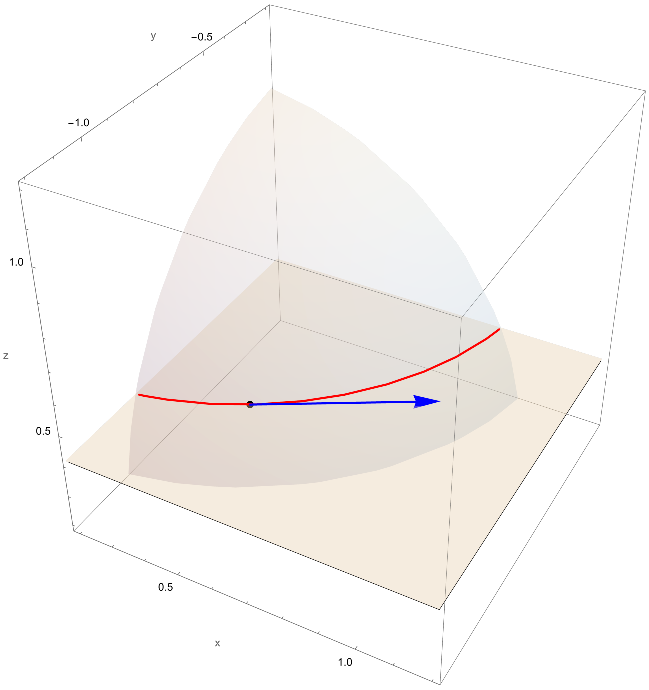

Figure 3.2: Illustration of optimization along the intersection of two constraint manifolds.

By gradient analysis, we obtain:

1. For any point on the surface $g(x) = 0$, $\nabla g(x)$ is orthogonal to the surface and points outward from the feasible region $g(x) \leq 0$. This is straightforward, since $\nabla g(x)$ indicates the steepest direction of increase.
2. For a local minimum point x^*:
   (a). If x^* lies on the surface $g(x) = 0$ (in other words, the constraint is *active*), then $-\nabla f(x^*)$ must be aligned with $\nabla g(x^*)$. Equivalently, $\exists\, \mu > 0$ such that
   $$\nabla f(x^*) + \mu \nabla g(x^*) = 0$$
   (b). If x^* does not lie on the surface, i.e., $g(x) < 0$ (the constraint is *inactive*), we simply require $\nabla f(x^*) = 0$. In the unified form above, this corresponds to $\exists\, \mu = 0$ such that
   $$\nabla f(x^*) + \mu \nabla g(x^*) = 0$$

Note *How should we understand the meaning of an "active" constraint? When the inequality constraint is not tight (i.e., $g(x) < 0$), it does not affect gradient descent optimization. In this case, the constraint is effectively absent, which is why we call it* inactive.

Again, we define the Lagrangian as

$$L(x, \mu) = f(x) + \mu g(x) \tag{3.9}$$

where $g(x)$ denotes the inequality constraint function. Following the same steps as in the equality-constrained case, we obtain:

Theorem 3.2

If x^* is a local minimum point, then

$$\begin{cases} \nabla L(x^*, \mu) = 0, \\ g(x^*) \leq 0, \\ \mu \geq 0, \\ \mu \cdot g(x^*) = 0. \end{cases} \tag{3.10}$$

The last condition (the *complementary slackness*) unifies the boundary and interior cases, since either $g(x^*) = 0$

(active constraint) or $\mu = 0$ (inactive constraint) must hold.

3.3 General Case

In the most general case, we consider the optimization problem

$$\operatorname*{argmin}_{x} f(x) \tag{3.11}$$

subject to equality constraints $h_i(x) = 0,\ i \in [k]$, and inequality constraints $g_j(x) \leq 0,\ j \in [l]$.

The corresponding Lagrangian is defined as

$$L(x, \lambda, \mu) := f(x) + \sum_{i \in [k]} \lambda_i h_i(x) + \sum_{j \in [l]} \mu_j g_j(x) \tag{3.12}$$

In this setting, a local minimum x^* must satisfy a series of conditions known as the **Karush–Kuhn–Tucker (KKT) conditions**:

Theorem 3.3 (KKT Conditions)

If x^* is a local minimum point, then there exist multipliers (λ, μ) such that

$$\begin{cases} \nabla L(x^*, \lambda, \mu) = 0, & \\ h_i(x^*) = 0, & \forall i \in [k], \\ g_j(x^*) \leq 0, & \forall j \in [l], \\ \mu_j \geq 0, & \forall j \in [l], \\ \mu_j \cdot g_j(x^*) = 0, & \forall j \in [l]. \end{cases} \tag{3.13}$$

♡

In most cases, the above conditions cannot be solved explicitly in closed form.

3.4 Support Vector Machine

3.4.1 Hard Margin

Consider a linearly separable dataset $\{(x_i, y_i)\}$ with $y_i \in \{0, 1\}$, and a linear model

$$f(x) = w^\top x + b \tag{3.14}$$

Although introducing a regularization term can make the solution unique by adjusting the hyperparameter λ, this does not resolve the issue of determining which solution $f(x; \lambda)$ is preferable among all possible choices, since λ is itself arbitrary.

Our goal is for the model to generalize well to test data drawn from the same distribution as the training set. A direct and intuitive way to achieve this is to choose the separating hyperplane that maximizes the distance between the plane and the nearest datapoints.

Definition 3.1 (Support Vectors)

The datapoints that lie closest to the separating hyperplane, such that the vectors orthogonal to the hyperplane connect the hyperplane to these datapoints, are called support vectors.

♣

In the case of SVM, we aim to find the **Max-Margin Classifier**, i.e., the hyperplane that maximizes the minimum margin across all training points. Here, the *margin* refers to the distance from a point to the hyperplane, which is equivalently the norm of the vector associated with its support vector.

Remark Why does $w^\top x + b = 0$ represent a hyperplane? Consider the hyperplane through the origin, given by $w^\top x = 0$. Translating this plane until it passes through some point $x_0 \notin \{x \mid w^\top x = 0\}$, we obtain

$$w^\top (x - x_0) = 0$$

Setting $-w^\top x_0 = b$, we recover the general hyperplane equation

$$w^\top x + b = 0$$

Remark Note that the interpretation of b differs between the two cases $y = w^\top x + b$ and $0 = w^\top x + b$. In the latter, we can observe that $|b| = \|w\|\|x_0\|$, which corresponds to the offset determined by x_0.

The distance between a datapoint and the hyperplane is given by:

Theorem 3.4

$$d = \frac{|w^\top x + b|}{\|w\|}. \tag{3.15}$$

♡

Geometrically, we obtain:

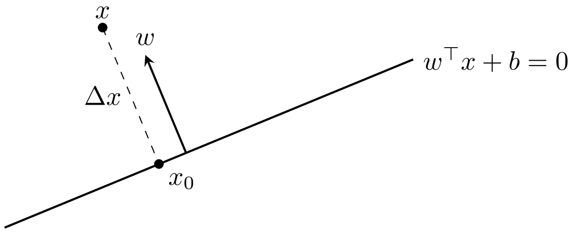

Figure 3.3: Support Vector Machine.

1. Consider

$$\begin{cases} x_0 + \Delta x \dfrac{w}{\|w\|} = x, \\ w^\top x_0 + b = 0, \end{cases}$$

so that

$$\begin{aligned} w^\top x_0 + w^\top \Delta x \frac{w}{\|w\|} &= w^\top x \\ -b + w^\top \Delta x \frac{w}{\|w\|} &= w^\top x \\ \Delta x \|w\| &= w^\top x + b \end{aligned}$$

This matches exactly the expression we obtained in (3.15).

2. From the Lagrangian perspective:

$$\begin{aligned} &\operatorname*{argmin}_{x_0} \frac{1}{2}\|x - x_0\|^2 &\Rightarrow\quad & L(x_0, \lambda) = \frac{1}{2}\|x - x_0\|^2 + \lambda(w^\top x_0 + b) \\ &s.t.\ w^\top x_0 + b & & \nabla_{x_0} L(x_0, \lambda) = 0 \quad\Rightarrow\quad x_0 - x + \lambda w = 0 \\ & & & \nabla_{x} L(x_0, \lambda) = 0 \quad\Rightarrow\quad w^\top x_0 + b = 0 \end{aligned}$$

This leads to the same solution as above.

When $\Delta > 0$, the point x lies on the positive side of the hyperplane; when $\Delta = 0$, x lies exactly on the hyperplane; and when $\Delta < 0$, x lies on the negative side.

We define

$$\gamma_i = \frac{y_i(w^\top x_i + b)}{\|w\|} \tag{3.16}$$

to incorporate both the label information and the margin (distance to the hyperplane). If $\gamma_i > 0$, the point x_i is correctly classified.

Define the margin

$$\gamma = \min_{i \in [n]} \gamma_i \tag{3.17}$$

and the SVM optimization problem can be formalized as

$$\underset{w,b}{\operatorname{argmax}}\ \gamma, \quad \text{s.t. } \forall i,\ \frac{y_i(w^\top x_i + b)}{\|w\|} \geq \gamma \tag{3.18}$$

This formulation seems problematic, since γ is not independent. To resolve this, suppose (x_0, y_0) is a point achieving the minimum margin γ_i. Then

$$\gamma = \frac{y_0(w^\top x_0 + b)}{\|w\|} \tag{3.19}$$

Thus, the SVM objective can be reformulated as

$$\underset{w,b}{\operatorname{argmax}}\ \frac{y_0(w^\top x_0 + b)}{\|w\|}, \quad \text{s.t. } \forall i,\ y_i(w^\top x_i + b) \geq y_0(w^\top x_0 + b). \tag{3.20}$$

We claim that we can make $y_0(w^\top x_0 + b)$ arbitrarily large without affecting the result.

Remark This is immediate: scaling w and b by the same factor λ does not change the optimization, since the numerator $y_i(w^\top x_i + b)$ and the denominator $\|w\|$ both scale by λ, and the factor cancels during simplification.

Thus, we may set $y_0(w^\top x_0 + b) = 1$. Define the *functional margin* as $y_i(w^\top x_i + b)$ and the *geometric margin* as $y_i(w^\top x_i + b)/\|w\|$. Since the functional margin alone has no intrinsic meaning, it is natural to rescale it by setting the norm of the support vector to unity.

This yields the primal form of the SVM optimization problem:

$$\underset{w,b}{\operatorname{argmax}}\ \frac{1}{\|w\|}, \quad \text{s.t. } y_i(w^\top x_i + b) \geq 1 \quad \forall i \tag{3.21}$$

which is equivalent to

$$\underset{w,b}{\operatorname{argmin}}\ \frac{1}{2}\|w\|^2, \quad \text{s.t. } y_i(w^\top x_i + b) \geq 1 \quad \forall i \tag{3.22}$$

In other words, the goal is to separate the dataset correctly while ensuring a sufficient margin, and simultaneously minimizing the norm of w. This principle is known as **Structural Risk Minimization** (SRM).

Remark Unlike logistic regression, datapoint far from separate hyperplane won't contribute loss. That is, only small subset of dataset is active as a constraint.

Proposition 3.1

SVM is a convex quadratic programming, and we can solve it in poltnomial time with standard package. ♡

Problem 3.1

1. What if the dataset is not linearly separable?
2. Wht if γ too small because of outliers?