Chapter 1

Linear Regression

Ordinary Least Squares

Theorem 1.0.1 (Uniqueness of the Least Squares Solution)**.** *Let $\Phi \in \mathbb{R}^{N\times M}$ denote the design matrix and $t \in \mathbb{R}^N$ the target vector. Consider the least squares cost function*

$$E(w) = \frac{1}{2}\|t - \Phi w\|^2.$$

Then:

(i) The function $E(w)$ is convex in w.

(ii) If $\Phi^\top\Phi$ is invertible (i.e., $\mathrm{rank}(\Phi) = M$), then $E(w)$ is strictly convex *and admits a unique minimizer*

$$w^* = (\Phi^\top\Phi)^{-1}\Phi^\top t.$$

(iii) If $\Phi^\top\Phi$ is singular, the minimizer is not unique; all minimizers are of the form

$$w = w_0 + v, \qquad v \in \mathrm{Null}(\Phi),$$

where w_0 is any particular solution to the normal equations $\Phi^\top\Phi w = \Phi^\top t$.

Proof. We begin by expanding the objective:

$$E(w) = \frac{1}{2}(t - \Phi w)^\top(t - \Phi w) = \frac{1}{2}(t^\top t - 2t^\top\Phi w + w^\top\Phi^\top\Phi w).$$

(1) Gradient and Stationary Point: The gradient of $E(w)$ with respect to w is

$$\nabla_w E(w) = -\Phi^\top t + \Phi^\top\Phi w.$$

Setting $\nabla_w E(w) = 0$ yields the *normal equations*

$$\Phi^\top\Phi w = \Phi^\top t. \tag{1}$$

(2) Hessian and Convexity: The Hessian of $E(w)$ is

$$H = \nabla_w^2 E(w) = \Phi^\top\Phi.$$

For any nonzero vector $z \in \mathbb{R}^M$,

$$z^\top H z = z^\top\Phi^\top\Phi z = \|\Phi z\|^2 \geq 0,$$

hence H is positive semidefinite, implying $E(w)$ is convex.
If Φ has full column rank $(\mathrm{rank}(\Phi) = M)$, then $\Phi^\top\Phi$ is positive definite, and

$$z^\top H z = 0 \quad \Leftrightarrow \quad z = 0,$$

so $E(w)$ is strictly convex. A strictly convex function has a unique minimizer, obtained by solving (1):

$$w^* = (\Phi^\top\Phi)^{-1}\Phi^\top t.$$

(3) Non-uniqueness for Rank-Deficient Φ: If $\Phi^\top\Phi$ is singular, there exist nonzero vectors v such that $\Phi v = 0$. For any particular solution w_0 satisfying (1), we have

$$\Phi^\top\Phi(w_0 + v) = \Phi^\top\Phi w_0 + \Phi^\top\Phi v = \Phi^\top t,$$

since $\Phi v = 0$. Thus, every vector $w = w_0 + v$, with $v \in \mathrm{Null}(\Phi)$, minimizes $E(w)$. The minimal-norm solution among them is given by the Moore–Penrose pseudoinverse:

$$w^* = \Phi^+ t.$$

(4) Conclusion: The cost $E(w)$ is convex for all Φ, and strictly convex (hence uniquely minimized) iff $\Phi^\top\Phi$ is invertible. □

Theorem 1.0.2 (Unbiasedness of the OLS Estimator)**.** *Assume the linear regression model*

$$t = \Phi w + \varepsilon,$$

where $\Phi \in \mathbb{R}^{N\times M}$ is the design matrix, $w \in \mathbb{R}^M$ the true parameter vector, and the noise satisfies $\mathbb{E}[\varepsilon] = 0$ and $\mathrm{Cov}(\varepsilon) = \sigma^2 I$. Assume further that $\Phi^\top\Phi$ is invertible. Then the ordinary least squares estimator

$$\hat{w} = (\Phi^\top\Phi)^{-1}\Phi^\top t$$

is an unbiased estimator of w, i.e.

$$\mathbb{E}[\hat{w}] = w.$$

Proof. By the model,

$$t = \Phi w + \varepsilon.$$

Substitute into the estimator:

$$\hat{w} = (\Phi^\top \Phi)^{-1}\Phi^\top t = (\Phi^\top \Phi)^{-1}\Phi^\top (\Phi w + \varepsilon).$$

Distribute terms:

$$\hat{w} = (\Phi^\top \Phi)^{-1}\Phi^\top \Phi\, w \;+\; (\Phi^\top \Phi)^{-1}\Phi^\top \varepsilon.$$

Since $(\Phi^\top \Phi)^{-1}\Phi^\top \Phi = I_M$, this simplifies to

$$\hat{w} = w + (\Phi^\top \Phi)^{-1}\Phi^\top \varepsilon.$$

Take expectation using linearity and $\mathbb{E}[\varepsilon] = 0$:

$$\mathbb{E}[\hat{w}] = \mathbb{E}\big[w + (\Phi^\top \Phi)^{-1}\Phi^\top \varepsilon\big] = w + (\Phi^\top \Phi)^{-1}\Phi^\top \mathbb{E}[\varepsilon] = w + (\Phi^\top \Phi)^{-1}\Phi^\top\, 0 = w.$$

Thus $\hat{w}$ is unbiased. □

Corollary 1.0.3. *Under the same assumptions,*

$$\mathrm{Cov}(\hat{w}) = \sigma^2(\Phi^\top \Phi)^{-1}.$$

Proof. From $\hat{w} = w + (\Phi^\top \Phi)^{-1}\Phi^\top \varepsilon$ and $\mathrm{Cov}(\varepsilon) = \sigma^2 I$,

$$\mathrm{Cov}(\hat{w}) = (\Phi^\top \Phi)^{-1}\Phi^\top\, \mathrm{Cov}(\varepsilon)\, \Phi(\Phi^\top \Phi)^{-1} = \sigma^2(\Phi^\top \Phi)^{-1}\Phi^\top \Phi(\Phi^\top \Phi)^{-1} = \sigma^2(\Phi^\top \Phi)^{-1}.$$

□

Theorem 1.0.4 (Covariance of the OLS Estimator)**.** *Under the linear regression model*

$$t = \Phi w + \varepsilon, \qquad \mathbb{E}[\varepsilon] = 0, \quad \mathrm{Cov}(\varepsilon) = \sigma^2 I,$$

with $\Phi \in \mathbb{R}^{N\times M}$ of full column rank, the ordinary least squares estimator

$$\hat{w} = (\Phi^\top \Phi)^{-1}\Phi^\top t$$

has covariance matrix

$$\mathrm{Cov}(\hat{w}) = \sigma^2(\Phi^\top \Phi)^{-1}.$$

Proof. From the model $t = \Phi w + \varepsilon$,

$$\hat{w} = (\Phi^\top \Phi)^{-1}\Phi^\top t = (\Phi^\top \Phi)^{-1}\Phi^\top (\Phi w + \varepsilon) = w + (\Phi^\top \Phi)^{-1}\Phi^\top \varepsilon.$$

Subtract the expectation $\mathbb{E}[\hat{w}] = w$ to get the deviation:

$$\hat{w} - \mathbb{E}[\hat{w}] = (\Phi^\top \Phi)^{-1}\Phi^\top \varepsilon.$$

Now compute the covariance:

$$\begin{aligned}\mathrm{Cov}(\hat{w}) &= \mathbb{E}\big[(\hat{w} - \mathbb{E}[\hat{w}])(\hat{w} - \mathbb{E}[\hat{w}])^\top\big]\\ &= \mathbb{E}\big[(\Phi^\top \Phi)^{-1}\Phi^\top \varepsilon\varepsilon^\top \Phi(\Phi^\top \Phi)^{-1}\big].\end{aligned}$$

Using $\mathrm{Cov}(\varepsilon) = \sigma^2 I$ and the linearity of expectation:

$$\mathrm{Cov}(\hat{w}) = (\Phi^\top \Phi)^{-1}\Phi^\top (\sigma^2 I)\Phi(\Phi^\top \Phi)^{-1} = \sigma^2(\Phi^\top \Phi)^{-1}\Phi^\top \Phi(\Phi^\top \Phi)^{-1}.$$

Simplifying:

$$\boxed{\mathrm{Cov}(\hat{w}) = \sigma^2(\Phi^\top \Phi)^{-1}}.$$

□

Theorem 1.0.5 (Gauss–Markov Theorem)**.** *Consider the linear model*

$$t = \Phi w + \varepsilon,$$

with $\Phi \in \mathbb{R}^{N\times M}$ of full column rank, $\mathbb{E}[\varepsilon] = 0$, and $\mathrm{Cov}(\varepsilon) = \sigma^2 I$. Let $\hat{w}_{\mathrm{OLS}} = (\Phi^\top \Phi)^{-1}\Phi^\top t$ denote the ordinary least squares estimator. Then $\hat{w}_{\mathrm{OLS}}$ is the Best Linear Unbiased Estimator (BLUE)*: for any other linear unbiased estimator of the form $\tilde{w} = Ct$ (with constant matrix $C \in \mathbb{R}^{M\times N}$ such that $\mathbb{E}[\tilde{w}] = w$), we have*

$$\mathrm{Cov}(\tilde{w}) - \mathrm{Cov}(\hat{w}_{\mathrm{OLS}}) \;\succeq\; 0,$$

i.e. the matrix difference is positive semidefinite. Equivalently, every componentwise variance of $\tilde{w}$ is at least that of $\hat{w}_{\mathrm{OLS}}$.

Proof. Let $\tilde{w}$ be any linear estimator of the form $\tilde{w} = Ct$ for a fixed matrix $C \in \mathbb{R}^{M\times N}$. The unbiasedness condition $\mathbb{E}[\tilde{w}] = w$ requires

$$\mathbb{E}[Ct] = C\mathbb{E}[t] = C\Phi w = w \quad \text{for all } w,$$

hence

$$C\Phi = I_M. \tag{1}$$

Write the OLS estimator as

$$\hat{w} \equiv \hat{w}_{\mathrm{OLS}} = (\Phi^\top \Phi)^{-1}\Phi^\top t.$$

Define the matrix difference

$$A \;:=\; C - (\Phi^\top \Phi)^{-1}\Phi^\top.$$

Using (1) and the identity $\big((\Phi^\top \Phi)^{-1}\Phi^\top\big)\Phi = I_M$, we obtain

$$A\Phi = C\Phi - (\Phi^\top \Phi)^{-1}\Phi^\top \Phi = I_M - I_M = 0.$$

Thus

$$A\Phi = 0 \qquad \Longrightarrow \qquad A\Phi w = 0 \quad \text{for all } w.$$

Now express $\tilde{w}$ in terms of $\hat{w}$ and A:

$$\tilde{w} = Ct = \big((\Phi^\top \Phi)^{-1}\Phi^\top + A\big)t = \hat{w} + At.$$

Subtracting expectations (and using $\mathbb{E}[\hat{w}] = \mathbb{E}[\tilde{w}] = w$) gives the zero-mean deviations

$$\tilde{w} - w = (\hat{w} - w) + A\varepsilon,$$

since $t = \Phi w + \varepsilon$ and $A\Phi w = 0$.
Compute the covariance matrices. Using $\mathrm{Cov}(\varepsilon) = \sigma^2 I$ and independence of deterministic matrices from ε,

$$\begin{aligned}\mathrm{Cov}(\tilde{w}) &= \mathbb{E}\big[(\tilde{w} - w)(\tilde{w} - w)^\top\big]\\ &= \mathbb{E}\big[(\hat{w} - w + A\varepsilon)(\hat{w} - w + A\varepsilon)^\top\big]\\ &= \mathrm{Cov}(\hat{w}) + A\,\mathbb{E}[\varepsilon\varepsilon^\top]\,A^\top \;+\; \mathbb{E}\big[(\hat{w} - w)\varepsilon^\top\big]A^\top \;+\; A\,\mathbb{E}\big[\varepsilon(\hat{w} - w)^\top\big].\end{aligned}$$

But $\hat{w} - w = (\Phi^\top \Phi)^{-1} \Phi^\top \varepsilon$ is linear in ε, so

$$\mathbb{E}\big[(\hat{w} - w)\varepsilon^\top\big] = (\Phi^\top \Phi)^{-1} \Phi^\top \mathbb{E}[\varepsilon\varepsilon^\top] = (\Phi^\top \Phi)^{-1} \Phi^\top (\sigma^2 I) = \sigma^2 (\Phi^\top \Phi)^{-1} \Phi^\top.$$

Since $A\Phi = 0$, we have

$$\mathbb{E}\big[(\hat{w} - w)\varepsilon^\top\big] A^\top = \sigma^2 (\Phi^\top \Phi)^{-1} \Phi^\top A^\top = \sigma^2 (\Phi^\top \Phi)^{-1} (\Phi^\top A^\top) = \sigma^2 (\Phi^\top \Phi)^{-1} (A\Phi)^\top = 0.$$

Similarly the other cross term $A\,\mathbb{E}[\varepsilon(\hat{w} - w)^\top]$ vanishes. Thus the covariance simplifies to

$$\operatorname{Cov}(\tilde{w}) = \operatorname{Cov}(\hat{w}) + A\,\mathbb{E}[\varepsilon\varepsilon^\top]\,A^\top = \operatorname{Cov}(\hat{w}) + \sigma^2 AA^\top.$$

Therefore

$$\operatorname{Cov}(\tilde{w}) - \operatorname{Cov}(\hat{w}) = \sigma^2 AA^\top.$$

But $\sigma^2 AA^\top$ is positive semidefinite (for any $\sigma^2 \geq 0$ and any matrix A), so

$$\operatorname{Cov}(\tilde{w}) - \operatorname{Cov}(\hat{w}) \succeq 0,$$

which proves that $\hat{w}$ has the smallest covariance matrix among all linear unbiased estimators. This completes the proof. □

Theorem 1.0.6 (Orthogonality of Residuals). *Let $\Phi \in \mathbb{R}^{N \times M}$ be the design matrix and $t \in \mathbb{R}^N$ the observed targets. Let $\hat{w}$ be any solution of the normal equations*

$$\Phi^\top \Phi\, \hat{w} \;=\; \Phi^\top t.$$

Define the residual vector $r := t - \Phi\hat{w}$. Then

$$\Phi^\top r \;=\; 0,$$

i.e. r is orthogonal to every column of Φ (equivalently r is orthogonal to $\operatorname{col}(\Phi)$).

Proof. Starting from the normal equations,

$$\Phi^\top \Phi\, \hat{w} \;=\; \Phi^\top t.$$

Rearrange terms to move $\Phi^\top \Phi \hat{w}$ to the right-hand side:

$$\Phi^\top t - \Phi^\top \Phi\, \hat{w} \;=\; 0.$$

Factor $\Phi^\top$:

$$\Phi^\top (t - \Phi\hat{w}) \;=\; 0.$$

But $t - \Phi\hat{w}$ is exactly the residual vector r, hence

$$\Phi^\top r \;=\; 0.$$

This shows each column of Φ has zero inner product with r, i.e. $r \perp \operatorname{col}(\Phi)$. □

Corollary 1.0.7 (Hat Matrix and Residual Projection). *If Φ has full column rank and $\hat{w} = (\Phi^\top \Phi)^{-1} \Phi^\top t$, define the hat (projection) matrix*

$$P := \Phi(\Phi^\top \Phi)^{-1} \Phi^\top.$$

Then the fitted values are $\hat{t} = Pt$ and the residual satisfies

$$r = (I - P)t,$$

with $P^2 = P$ and $P^\top = P$. Consequently $(I - P)$ is the orthogonal projector onto $\operatorname{col}(\Phi)^\perp$, and r is the orthogonal projection of t onto that complement.

Proof. Using $\hat{w} = (\Phi^\top \Phi)^{-1} \Phi^\top t$ gives $\hat{t} = \Phi\hat{w} = \Phi(\Phi^\top \Phi)^{-1} \Phi^\top t = Pt$, so $r = t - \hat{t} = (I - P)t$. The identities $P^2 = P$ and $P^\top = P$ follow from straightforward algebra:

$$P^2 = \Phi(\Phi^\top \Phi)^{-1} \underbrace{\Phi^\top \Phi}_{=} (\Phi^\top \Phi)^{-1} \Phi^\top = P, \qquad P^\top = \big(\Phi(\Phi^\top \Phi)^{-1} \Phi^\top\big)^\top = \Phi(\Phi^\top \Phi)^{-1} \Phi^\top = P.$$

Thus P is an orthogonal projector onto $\operatorname{col}(\Phi)$ and $(I - P)$ projects orthogonally onto its complement, so r lies in $\operatorname{col}(\Phi)^\perp$. □

Bayesian Linear Regression: Prior on w and Predictive Distribution

Bayesian Formulation

In Bayesian linear regression we treat the parameter vector w as a random variable and place a prior distribution on it. The generative model is:

$$t = \Phi w + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \beta^{-1} I_N),$$

where β is the noise precision.

Prior Distribution on w

We choose a zero-mean isotropic Gaussian prior:

$$p(w) = \mathcal{N}(w \mid 0, \alpha^{-1} I_M),$$

where α is the prior precision. This encodes the belief that large weights are unlikely (acts as a regularizer).

Likelihood

Conditioned on w, the likelihood of the data is:

$$p(t \mid \Phi, w, \beta) = \mathcal{N}(t \mid \Phi w, \beta^{-1} I_N).$$

Posterior Distribution of w

By Bayes' theorem:

$$p(w \mid t, \Phi) \propto p(t \mid \Phi, w, \beta)\, p(w).$$

Because both prior and likelihood are Gaussian, the posterior is also Gaussian:

$$p(w \mid t, \Phi) = \mathcal{N}(w \mid m_N, S_N),$$

with posterior precision and covariance given by:

$$S_N^{-1} = \alpha I_M + \beta \Phi^\top \Phi, \qquad S_N = (\alpha I_M + \beta \Phi^\top \Phi)^{-1},$$

and the posterior mean:

$$m_N = \beta S_N \Phi^\top t.$$

Interpretation

- m_N is the Bayes estimate of w (posterior mean).
- S_N quantifies uncertainty in the weight estimates.
- As $\alpha \to 0$ (weak prior),

  $$m_N \to (\Phi^\top \Phi)^{-1} \Phi^\top t,$$

  recovering the ordinary least squares solution.

Predictive Distribution

For a new input x_* with feature vector $\phi_* = \phi(x_*)$, the predictive distribution integrates over the posterior uncertainty in w:

$$p(t_* \mid x_*, t, \Phi) = \int p(t_* \mid x_*, w, \beta)\, p(w \mid t, \Phi)\, dw.$$

The integrand is a product of two Gaussians, so the predictive distribution is Gaussian:

$$p(t_* \mid x_*, t, \Phi) = \mathcal{N}\big(t_* \mid m_N^\top \phi_*,\ \beta^{-1} + \phi_*^\top S_N \phi_*\big).$$

Predictive Mean and Variance

Predictive Mean:

$$\mathbb{E}[t_* \mid x_*, t, \Phi] = m_N^\top \phi_*.$$

Predictive Variance:

$$\mathrm{Var}(t_* \mid x_*, t, \Phi) = \underbrace{\beta^{-1}}_{\text{noise variance}} + \underbrace{\phi_*^\top S_N \phi_*}_{\text{model uncertainty}}.$$

Thus the predictive variance decomposes into:

- aleatoric noise (irreducible), and
- epistemic uncertainty (reduced with more data).

Likelihood Derivation (Gaussian Noise) and MLEs

1. Single-observation likelihood

Assume the data generation model for a single observation:

$$t_n = w^\top \phi(x_n) + \varepsilon_n, \qquad \varepsilon_n \sim \mathcal{N}(0, \beta^{-1}).$$

Then the conditional density (likelihood) for t_n given w is

$$p(t_n \mid x_n, w, \beta) = \mathcal{N}\big(t_n \mid w^\top \phi(x_n),\ \beta^{-1}\big) = \sqrt{\frac{\beta}{2\pi}}\, \exp\Big(-\tfrac{\beta}{2}\big(t_n - w^\top \phi(x_n)\big)^2\Big).$$

2. Joint likelihood for the dataset

Assuming i.i.d. noise, the joint likelihood for all N observations is the product

$$p(t \mid \Phi, w, \beta) = \prod_{n=1}^{N} p(t_n \mid x_n, w, \beta) = \left(\frac{\beta}{2\pi}\right)^{N/2} \exp\Big(-\tfrac{\beta}{2}\sum_{n=1}^{N}(t_n - w^\top \phi(x_n))^2\Big).$$

Using matrix notation with $\Phi \in \mathbb{R}^{N\times M}$ and $t \in \mathbb{R}^N$:

$$p(t \mid \Phi, w, \beta) = \left(\frac{\beta}{2\pi}\right)^{N/2} \exp\Big(-\tfrac{\beta}{2}\|t - \Phi w\|^2\Big).$$

3. Log-likelihood

The log-likelihood (more convenient for optimization) is

$$\ell(w, \beta) := \log p(t \mid \Phi, w, \beta) = \frac{N}{2}\log\beta - \frac{N}{2}\log(2\pi) - \frac{\beta}{2}\|t - \Phi w\|^2.$$

Dropping constants independent of the parameters when optimizing:

$$\ell(w, \beta) = \frac{N}{2}\log\beta - \frac{\beta}{2}\|t - \Phi w\|^2 + \text{const}.$$

4. MLE for w (given β)

Take gradient of the log-likelihood w.r.t. w:

$$\nabla_w \ell(w, \beta) = -\frac{\beta}{2}\cdot 2\,(-\Phi^\top)(t - \Phi w) = \beta\Phi^\top(t - \Phi w).$$

Set to zero for critical point:

$$\Phi^\top(t - \Phi w) = 0 \quad \Rightarrow \quad \Phi^\top\Phi\, w = \Phi^\top t.$$

If $\Phi^\top\Phi$ is invertible, the MLE of w is

$$\boxed{\hat{w}_{\text{MLE}} = (\Phi^\top\Phi)^{-1}\Phi^\top t}$$

which is the ordinary least squares solution. Thus MLE = least squares under Gaussian noise.

5. MLE for noise precision β (given w)

Differentiate ℓ w.r.t. β:

$$\frac{\partial \ell}{\partial \beta} = \frac{N}{2\beta} - \frac{1}{2}\|t - \Phi w\|^2.$$

Set equal to zero:

$$\frac{N}{2\beta} = \frac{1}{2}\|t - \Phi w\|^2 \quad \Rightarrow \quad \hat{\beta}_{\text{MLE}} = \frac{N}{\|t - \Phi w\|^2}.$$

If we substitute $w = \hat{w}_{\text{MLE}}$ we get the MLE for β:

$$\boxed{\hat{\beta}_{\text{MLE}} = \frac{N}{\|t - \Phi\hat{w}_{\text{MLE}}\|^2}}.$$

Equivalently, the MLE for noise variance $\sigma^2 = \beta^{-1}$ is

$$\hat{\sigma}^2_{\text{MLE}} = \frac{1}{N}\|t - \Phi\hat{w}_{\text{MLE}}\|^2.$$

(For an unbiased estimator of σ^2 divide by $N - M$ instead of N.)

6. Negative log-likelihood and connection to MAP

The negative log-likelihood (up to additive constant) is

$$-\ell(w, \beta) \propto \frac{\beta}{2}\|t - \Phi w\|^2 - \frac{N}{2}\log\beta.$$

When combining with a Gaussian prior $p(w) \propto \exp\big(-\frac{\alpha}{2}\|w\|^2\big)$, the negative log-posterior (up to constants) becomes

$$-\log p(w \mid t) \propto \frac{\beta}{2}\|t - \Phi w\|^2 + \frac{\alpha}{2}\|w\|^2,$$

whose minimizer yields the MAP estimator. Dividing through by β and setting $\lambda = \alpha/\beta$ gives the familiar ridge form:

$$\hat{w}_{\text{MAP}} = (\Phi^\top\Phi + \lambda I)^{-1}\Phi^\top t.$$

Derivation of the Posterior with a Gaussian Prior (Completing the Square)

Assume the Gaussian likelihood and Gaussian prior:

$$p(t \mid w) \propto \exp\Big(-\tfrac{\beta}{2}\|t-\Phi w\|^2\Big), \qquad p(w) \propto \exp\Big(-\tfrac{\alpha}{2}\|w\|^2\Big).$$

Posterior (unnormalized) by Bayes' rule:

$$p(w \mid t) \propto p(t \mid w)\, p(w) \propto \exp\Big(-\tfrac{\beta}{2}\|t-\Phi w\|^2 - \tfrac{\alpha}{2}\|w\|^2\Big).$$

Expand the exponents (quadratic form in w).

$$\begin{aligned} &\tfrac{\beta}{2}\|t-\Phi w\|^2 + \tfrac{\alpha}{2}\|w\|^2 \\ &= \tfrac{\beta}{2}\big(t^\top t - 2t^\top \Phi w + w^\top \Phi^\top \Phi w\big) + \tfrac{\alpha}{2} w^\top w \\ &= \tfrac{1}{2}\, w^\top(\beta \Phi^\top \Phi + \alpha I)\, w \;-\; \beta t^\top \Phi w + \tfrac{\beta}{2} t^\top t. \end{aligned}$$

Group terms in w and complete the square. Write the quadratic form as

$$\tfrac{1}{2}\, w^\top A\, w - b^\top w + \text{const}, \quad \text{where } A = \beta\Phi^\top\Phi + \alpha I, \quad b = \beta \Phi^\top t.$$

Complete the square:

$$\tfrac{1}{2} w^\top A w - b^\top w = \tfrac{1}{2}(w - A^{-1}b)^\top A (w - A^{-1}b) - \tfrac{1}{2} b^\top A^{-1} b.$$

Thus the unnormalized posterior becomes

$$p(w \mid t) \propto \exp\Big(-\tfrac{1}{2}(w - A^{-1}b)^\top A (w - A^{-1}b)\Big) \cdot \exp\Big(\tfrac{1}{2} b^\top A^{-1} b - \tfrac{\beta}{2} t^\top t\Big).$$

The second exponential is independent of w and becomes part of the normalizing constant.

Identify posterior covariance and mean. Hence the posterior is Gaussian with precision A and covariance $S_N = A^{-1}$:

$$S_N = (\beta \Phi^\top \Phi + \alpha I)^{-1},$$

and posterior mean

$$m_N = A^{-1} b = (\beta\Phi^\top\Phi + \alpha I)^{-1}(\beta \Phi^\top t).$$

Simplify using $\lambda = \alpha/\beta$. Dividing numerator and denominator by β gives the more familiar form:

$$S_N = \beta^{-1}(\Phi^\top\Phi + \lambda I)^{-1}, \qquad m_N = (\Phi^\top \Phi + \lambda I)^{-1}\Phi^\top t,$$

where $\lambda = \alpha/\beta$. Note that m_N equals the ridge/MAP estimator and S_N quantifies posterior uncertainty.

Log-Likelihood and Log-Prior in Bayesian Linear Regression

Model Setup

We observe data $(\mathbf{X}, \mathbf{y})$ where $\mathbf{X} \in \mathbb{R}^{N\times D}$, $\mathbf{y} \in \mathbb{R}^N$ and weights $\mathbf{w} \in \mathbb{R}^D$. The linear-Gaussian model assumes

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}\big(0,\ \sigma^2 \mathbf{I}_N\big).$$

Log-Likelihood $\log p(\mathbf{y} \mid \mathbf{X}, \mathbf{w})$

Because the noise is i.i.d. Gaussian,

$$p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) = \mathcal{N}\big(\mathbf{y} \mid \mathbf{X}\mathbf{w},\ \sigma^2 \mathbf{I}_N\big).$$

Using the multivariate Gaussian density,

$$p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) = \frac{1}{(2\pi)^{N/2}\sigma^N} \exp\left(-\frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\mathbf{w})^\top (\mathbf{y} - \mathbf{X}\mathbf{w})\right).$$

Thus the log-likelihood is

$$\log p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) = -\frac{N}{2}\log(2\pi) - \frac{N}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w}).$$

Equivalently,

$$\log p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) = -\frac{N}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - \mathbf{x}_i^\top \mathbf{w})^2.$$

Log-Prior $\log p(\mathbf{w})$

Assume a zero-mean Gaussian prior

$$p(\mathbf{w}) = \mathcal{N}\big(\mathbf{w} \mid 0,\ \alpha^{-1}\mathbf{I}_D\big).$$

The density is

$$p(\mathbf{w}) = \left(\frac{\alpha}{2\pi}\right)^{D/2} \exp\Big(-\frac{\alpha}{2}\,\mathbf{w}^\top \mathbf{w}\Big).$$

Therefore the log-prior is

$$\log p(\mathbf{w}) = \frac{D}{2}\log(\alpha) - \frac{D}{2}\log(2\pi) - \frac{\alpha}{2}\mathbf{w}^\top\mathbf{w}.$$

MAP Estimation (Posterior Mode)

The posterior satisfies

$$p(\mathbf{w} \mid \mathbf{y}, \mathbf{X}) \propto p(\mathbf{y} \mid \mathbf{X}, \mathbf{w})\, p(\mathbf{w}).$$

Hence,

$$\log p(\mathbf{w} \mid \mathbf{y}, \mathbf{X}) = \log p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) + \log p(\mathbf{w}) + \text{const}.$$

Ignoring constants w.r.t. $\mathbf{w}$,

$$\log p(\mathbf{w} \mid \mathbf{y}, \mathbf{X}) = -\frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) - \frac{\alpha}{2}\mathbf{w}^\top\mathbf{w} + \text{const}.$$

Maximizing the posterior is equivalent to minimizing

$$(\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) \;+\; (\alpha\sigma^2)\,\mathbf{w}^\top\mathbf{w}.$$

Letting $\lambda = \alpha\sigma^2$, the MAP solution is the ridge-regression estimator

$$\mathbf{w}_{\text{MAP}} = \arg\min_{\mathbf{w}} \Big[\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \lambda \|\mathbf{w}\|^2 \Big].$$

LASSO (L1) as a MAP Estimate

We show that a Laplace prior on the weights leads to L1 regularization (LASSO).

1. The Laplace Prior Distribution

Assume a Laplace prior on the weights $\mathbf{w}$, which encourages sparsity (many weights at zero). We assume each weight w_j is drawn independently:

$$p(w_j) = \mathsf{Laplace}(w_j \mid 0, b) = \frac{1}{2b} \exp\left(-\frac{|w_j|}{b}\right)$$

The full prior for the D-dimensional vector $\mathbf{w}$ is the product:

$$p(\mathbf{w}) = \prod_{j=1}^{D} p(w_j) = \left(\frac{1}{2b}\right)^{D} \exp\left(-\frac{1}{b}\sum_{j=1}^{D} |w_j|\right)$$

This can be written using the L1-norm, $\|\mathbf{w}\|_1 = \sum_{j=1}^{D} |w_j|$:

$$p(\mathbf{w}) = \left(\frac{1}{2b}\right)^{D} \exp\left(-\frac{1}{b}\|\mathbf{w}\|_1\right)$$

2. The Log-Prior

Taking the natural logarithm to get the log-prior:

$$\begin{aligned}
\log p(\mathbf{w}) &= \log\left[\left(\frac{1}{2b}\right)^{D} \exp\left(-\frac{1}{b}\|\mathbf{w}\|_1\right)\right] \\
&= D\log\left(\frac{1}{2b}\right) - \frac{1}{b}\|\mathbf{w}\|_1 \\
&= \mathsf{const} - \frac{1}{b}\|\mathbf{w}\|_1
\end{aligned}$$

3. MAP Estimation

The MAP estimate maximizes the log-posterior, which is the sum of the log-likelihood and the log-prior:

$$\log p(\mathbf{w} \mid \mathbf{y}, \mathbf{X}) = \log p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) + \log p(\mathbf{w}) + \mathsf{const}$$

Substituting the Gaussian log-likelihood (from Section 7) [cite: 108] and the Laplace log-prior, ignoring all terms that are constant w.r.t. $\mathbf{w}$:

$$\log p(\mathbf{w} \mid \mathbf{y}, \mathbf{X}) \propto -\frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) - \frac{1}{b}\|\mathbf{w}\|_1$$

Maximizing this is equivalent to minimizing its negative. Using your defined ‘$\arg\min$‘ command:

$$\mathbf{w}_{\mathsf{MAP}} = \arg\min_{\mathbf{w}}\left[\frac{1}{2\sigma^2}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \frac{1}{b}\|\mathbf{w}\|_1\right]$$

Multiplying by the constant $2\sigma^2$ and defining $\lambda = \frac{2\sigma^2}{b}$ gives the familiar LASSO objective function:

$$\boxed{\mathbf{w}_{\mathsf{MAP}} = \arg\min_{\mathbf{w}}\left[\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \lambda\|\mathbf{w}\|_1\right]}$$

Generalized Least Squares (GLS)

1. The OLS Assumption vs. The GLS Model

The OLS estimator $\hat{w}_{\mathrm{OLS}}$ is the BLUE (Best Linear Unbiased Estimator) under the Gauss-Markov assumptions, which critically require that the error covariance matrix is *spherical*:

$$\mathrm{Cov}(\varepsilon) = \sigma^2 I_N$$

This single assumption implies two conditions:

- **Homoscedasticity:** All errors have the same variance σ^2.
- **No Autocorrelation:** All errors are uncorrelated.

In the **Generalized Least Squares (GLS)** model, we relax this assumption. We assume the errors are still zero-mean but have a general, known, $N \times N$ positive-definite covariance matrix $\mathbf{\Sigma}$:

$$\mathbb{E}[\varepsilon] = 0, \qquad \mathrm{Cov}(\varepsilon) = \mathbf{\Sigma}$$

When $\mathbf{\Sigma} \neq \sigma^2 I_N$, the OLS estimator $\hat{w}_{\mathrm{OLS}}$ is still *unbiased*, but it is no longer BLUE (i.e., it is not the minimum-variance estimator).

2. Derivation via Data Whitening

The core idea of GLS is to transform the generalized model back into a ”standard” model that satisfies the OLS assumptions. This is done via *whitening*.

Since $\mathbf{\Sigma}$ is positive-definite, we can find a non-singular $N \times N$ matrix $\mathbf{C}$ such that $\mathbf{\Sigma} = \mathbf{C}\mathbf{C}^\top$ (e.g., via Cholesky decomposition). The inverse $\mathbf{C}^{-1}$ is our ”whitening” matrix.

Start with the original model:

$$t = \Phi w + \varepsilon$$

Pre-multiply by $\mathbf{C}^{-1}$:

$$(\mathbf{C}^{-1}t) = (\mathbf{C}^{-1}\Phi)w + (\mathbf{C}^{-1}\varepsilon)$$

Let's define our transformed variables:

$$\tilde{t} = \mathbf{C}^{-1}t, \qquad \tilde{\Phi} = \mathbf{C}^{-1}\Phi, \qquad \tilde{\varepsilon} = \mathbf{C}^{-1}\varepsilon$$

Our new, transformed model is:

$$\tilde{t} = \tilde{\Phi}w + \tilde{\varepsilon}$$

Now, let's find the covariance of the *new* error term $\tilde{\varepsilon}$:

$$\begin{aligned}
\mathrm{Cov}(\tilde{\varepsilon}) &= \mathrm{Cov}(\mathbf{C}^{-1}\varepsilon) \\
&= \mathbf{C}^{-1}\mathrm{Cov}(\varepsilon)(\mathbf{C}^{-1})^\top \\
&= \mathbf{C}^{-1}\mathbf{\Sigma}(\mathbf{C}^\top)^{-1} \\
&= \mathbf{C}^{-1}(\mathbf{C}\mathbf{C}^\top)(\mathbf{C}^\top)^{-1} \\
&= (\mathbf{C}^{-1}\mathbf{C})(\mathbf{C}^\top(\mathbf{C}^\top)^{-1}) = I_N \cdot I_N = I_N
\end{aligned}$$

The transformed model $\tilde{t} = \tilde{\Phi}w + \tilde{\varepsilon}$ has spherical errors ($\sigma^2 = 1$). It satisfies the Gauss-Markov assumptions!

3. The GLS (Aitken) Estimator

We can find the BLUE for w by simply applying the OLS formula to our *transformed* data:

$$\hat{w}_{\mathrm{GLS}} = (\tilde{\Phi}^\top\tilde{\Phi})^{-1}\tilde{\Phi}^\top\tilde{t}$$

Now, substitute the original variables back in.

- $\tilde{\Phi}^\top\tilde{\Phi} = (\mathbf{C}^{-1}\Phi)^\top(\mathbf{C}^{-1}\Phi) = \Phi^\top(\mathbf{C}^{-1})^\top\mathbf{C}^{-1}\Phi = \Phi^\top(\mathbf{C}\mathbf{C}^\top)^{-1}\Phi = \Phi^\top\mathbf{\Sigma}^{-1}\Phi$
- $\tilde{\Phi}^\top\tilde{t} = (\mathbf{C}^{-1}\Phi)^\top(\mathbf{C}^{-1}t) = \Phi^\top(\mathbf{C}^{-1})^\top\mathbf{C}^{-1}t = \Phi^\top\mathbf{\Sigma}^{-1}t$

Substituting these gives the **GLS estimator**:

$$\boxed{\hat{w}_{\text{GLS}} = (\Phi^\top\mathbf{\Sigma}^{-1}\Phi)^{-1}\Phi^\top\mathbf{\Sigma}^{-1}t}$$

This is also called the **Aitken estimator**.

4. Properties of the GLS Estimator

Theorem 1.1.1 (Aitken Theorem)**.** *Under the generalized model $t = \Phi w + \varepsilon$ with $\operatorname{Cov}(\varepsilon) = \mathbf{\Sigma}$:*

(i) **Unbiasedness:** *The GLS estimator is unbiased.*

$$\mathbb{E}[\hat{w}_{\text{GLS}}] = w$$

(ii) **Covariance:** *The covariance matrix of $\hat{w}_{\text{GLS}}$ is:*

$$\boxed{\operatorname{Cov}(\hat{w}_{\text{GLS}}) = (\Phi^\top\mathbf{\Sigma}^{-1}\Phi)^{-1}}$$

(iii) **Efficiency:** *$\hat{w}_{\text{GLS}}$ is the* **BLUE** *(Best Linear Unbiased Estimator). Any other linear unbiased estimator $\tilde{w}$ will have a larger covariance.*

5. OLS as a Special Case of GLS

If the OLS assumptions were correct, $\mathbf{\Sigma} = \sigma^2 I_N$. Let's plug this into the GLS formula:

$$\begin{aligned}
\hat{w}_{\text{GLS}} &= (\Phi^\top(\sigma^2 I_N)^{-1}\Phi)^{-1}\Phi^\top(\sigma^2 I_N)^{-1}t \\
&= (\Phi^\top(\tfrac{1}{\sigma^2})\Phi)^{-1}\Phi^\top(\tfrac{1}{\sigma^2})t \\
&= \left(\tfrac{1}{\sigma^2}(\Phi^\top\Phi)\right)^{-1}\left(\tfrac{1}{\sigma^2}\Phi^\top t\right) \\
&= \left(\sigma^2(\Phi^\top\Phi)^{-1}\right)\left(\tfrac{1}{\sigma^2}\Phi^\top t\right) \\
&= (\Phi^\top\Phi)^{-1}\Phi^\top t = \hat{w}_{\text{OLS}}
\end{aligned}$$

This confirms that OLS is just a special case of GLS where the error covariance is spherical.

Note 1.1.2. **Feasible GLS (FGLS):** In practice, the exact covariance $\mathbf{\Sigma}$ is almost never known. **FGLS** is the practical approach where $\mathbf{\Sigma}$ is *estimated* from the data (often using the residuals from an initial OLS fit). The $\hat{\mathbf{\Sigma}}$ is then plugged into the GLS formula.

Derivation of GLS as an MLE

The GLS estimator is the MLE for a linear model where the noise is drawn from a single multivariate Gaussian distribution, allowing for both heteroscedasticity and autocorrelation.

1. Probabilistic Model & Error Function Assume the linear model in vector form:

$$\mathbf{t} = \mathbf{\Phi}\mathbf{w} + \boldsymbol{\epsilon}$$

where the entire $N \times 1$ noise vector $\boldsymbol{\epsilon}$ is drawn from a zero-mean multivariate Gaussian with a general $N \times N$ positive-definite covariance matrix $\mathbf{\Sigma}$:

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$$

This implies the likelihood for the entire target vector $\mathbf{t}$ is:

$$p(\mathbf{t} \mid \mathbf{\Phi}, \mathbf{w}, \mathbf{\Sigma}) = \mathcal{N}(\mathbf{t} \mid \mathbf{\Phi}\mathbf{w}, \mathbf{\Sigma})$$

The probability density function (PDF) is:

$$p(\mathbf{t} \mid \dots) = \frac{1}{(2\pi)^{N/2}|\mathbf{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{t} - \mathbf{\Phi}\mathbf{w})^\top\mathbf{\Sigma}^{-1}(\mathbf{t} - \mathbf{\Phi}\mathbf{w})\right)$$

The log-likelihood $\mathcal{L}(\mathbf{w})$ is:

$$\mathcal{L}(\mathbf{w}) = \log p(\mathbf{t} \mid \dots) = C - \frac{1}{2}(\mathbf{t} - \mathbf{\Phi}\mathbf{w})^\top\mathbf{\Sigma}^{-1}(\mathbf{t} - \mathbf{\Phi}\mathbf{w})$$

where $C = -\frac{N}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{\Sigma}|$ is a constant with respect to $\mathbf{w}$.

To find the MLE, we maximize $\mathcal{L}(\mathbf{w})$, which is equivalent to minimizing the negative of the $\mathbf{w}$-dependent part. This gives the GLS error function $E(\mathbf{w})$:

$$E(\mathbf{w}) = (\mathbf{t} - \mathbf{\Phi}\mathbf{w})^\top\mathbf{\Sigma}^{-1}(\mathbf{t} - \mathbf{\Phi}\mathbf{w})$$

This quadratic form is the (generalized) squared Mahalanobis distance.

2. Derivation of the Closed-Form Solution The error function $E(\mathbf{w})$ is already in its matrix form. To find the minimum, we expand the expression. Let $\mathbf{\Omega} = \mathbf{\Sigma}^{-1}$ for simplicity.

$$E(\mathbf{w}) = (\mathbf{t}^\top - \mathbf{w}^\top\mathbf{\Phi}^\top)\mathbf{\Omega}(\mathbf{t} - \mathbf{\Phi}\mathbf{w})$$

$$E(\mathbf{w}) = \mathbf{t}^\top\mathbf{\Omega}\mathbf{t} - \mathbf{t}^\top\mathbf{\Omega}\mathbf{\Phi}\mathbf{w} - \mathbf{w}^\top\mathbf{\Phi}^\top\mathbf{\Omega}\mathbf{t} + \mathbf{w}^\top\mathbf{\Phi}^\top\mathbf{\Omega}\mathbf{\Phi}\mathbf{w}$$

Since $\mathbf{\Sigma}$ is symmetric, its inverse $\mathbf{\Omega}$ is also symmetric ($\mathbf{\Omega}^\top = \mathbf{\Omega}$). The middle terms are transposes of each other:

$$E(\mathbf{w}) = \mathbf{t}^\top\mathbf{\Omega}\mathbf{t} - 2\mathbf{w}^\top\mathbf{\Phi}^\top\mathbf{\Omega}\mathbf{t} + \mathbf{w}^\top(\mathbf{\Phi}^\top\mathbf{\Omega}\mathbf{\Phi})\mathbf{w}$$

Now, we take the gradient with respect to $\mathbf{w}$:

$$\nabla_{\mathbf{w}}E(\mathbf{w}) = -2\mathbf{\Phi}^\top\mathbf{\Omega}\mathbf{t} + 2(\mathbf{\Phi}^\top\mathbf{\Omega}\mathbf{\Phi})\mathbf{w}$$

Set the gradient to zero to find the minimum:

$$\mathbf{0} = -2\mathbf{\Phi}^\top\mathbf{\Omega}\mathbf{t} + 2(\mathbf{\Phi}^\top\mathbf{\Omega}\mathbf{\Phi})\mathbf{w}$$

$$(\mathbf{\Phi}^\top\mathbf{\Omega}\mathbf{\Phi})\mathbf{w} = \mathbf{\Phi}^\top\mathbf{\Omega}\mathbf{t}$$

Substituting back $\mathbf{\Omega} = \mathbf{\Sigma}^{-1}$, we get the **GLS Normal Equations**:

$$(\mathbf{\Phi}^\top\mathbf{\Sigma}^{-1}\mathbf{\Phi})\mathbf{w} = \mathbf{\Phi}^\top\mathbf{\Sigma}^{-1}\mathbf{t}$$

Assuming $(\mathbf{\Phi}^\top\mathbf{\Sigma}^{-1}\mathbf{\Phi})$ is invertible, we solve for $\mathbf{w}$ to get the GLS solution:

$$\boxed{\hat{\mathbf{w}}_{\text{GLS}} = (\mathbf{\Phi}^\top\mathbf{\Sigma}^{-1}\mathbf{\Phi})^{-1}\mathbf{\Phi}^\top\mathbf{\Sigma}^{-1}\mathbf{t}}$$

Weighted Least Squares (WLS)

WLS is a special case of GLS used when errors are heteroscedastic but uncorrelated.

1. The WLS Model and Objective

We assume the general linear model $t = \Phi w + \varepsilon$, where $\mathbb{E}[\varepsilon] = 0$ but the errors are heteroscedastic:

$$\mathrm{Cov}(\varepsilon) = \mathbf{\Sigma} = \begin{pmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_N^2 \end{pmatrix}$$

The WLS objective is to minimize the **Weighted Sum of Squared Residuals (WSSR)**, where each squared residual is weighted by the inverse of its variance, $w_i = 1/\sigma_i^2$.

$$E(w) = \sum_{i=1}^{N} w_i (t_i - \phi_i^\top w)^2$$

In matrix form, we define the diagonal weight matrix $\mathbf{W} = \mathbf{\Sigma}^{-1}$:

$$\mathbf{W} = \mathrm{diag}(w_1, \dots, w_N) = \mathrm{diag}(1/\sigma_1^2, \dots, 1/\sigma_N^2)$$

The objective function becomes:

$$E(w) = (t - \Phi w)^\top \mathbf{W} (t - \Phi w)$$

2. Derivation of the WLS Estimator

We find the estimator $\hat{w}_{\mathrm{WLS}}$ by minimizing $E(w)$. First, expand the objective:

$$E(w) = t^\top \mathbf{W} t - t^\top \mathbf{W} \Phi w - w^\top \Phi^\top \mathbf{W} t + w^\top \Phi^\top \mathbf{W} \Phi w$$

Since $w^\top \Phi^\top \mathbf{W} t$ is a scalar, it equals its transpose $t^\top \mathbf{W} \Phi w$.

$$E(w) = t^\top \mathbf{W} t - 2t^\top \mathbf{W} \Phi w + w^\top \Phi^\top \mathbf{W} \Phi w$$

Now, take the gradient with respect to w and set to zero:

$$\nabla_w E(w) = -2\Phi^\top \mathbf{W} t + 2\Phi^\top \mathbf{W} \Phi w$$

$$\nabla_w E(w) = 0 \quad \Rightarrow \quad 2\Phi^\top \mathbf{W} \Phi w = 2\Phi^\top \mathbf{W} t$$

This gives the **WLS Normal Equations**:

$$(\Phi^\top \mathbf{W} \Phi) w = \Phi^\top \mathbf{W} t$$

Assuming $\Phi^\top \mathbf{W} \Phi$ is invertible, the WLS estimator is:

$$\boxed{\hat{w}_{\mathrm{WLS}} = (\Phi^\top \mathbf{W} \Phi)^{-1} \Phi^\top \mathbf{W} t}$$

This is identical to the GLS estimator where $\mathbf{\Sigma}^{-1} = \mathbf{W}$.

3. Derivation via Data Whitening

We can also derive WLS by transforming the data so that the new error terms are homoscedastic with variance 1, and then applying OLS. Let $\mathbf{W}^{1/2}$ be the diagonal matrix with entries $\sqrt{w_i} = 1/\sigma_i$.

$$\mathbf{W}^{1/2} = \mathrm{diag}(1/\sigma_1, \dots, 1/\sigma_N)$$

Pre-multiply the original model $t = \Phi w + \varepsilon$ by $\mathbf{W}^{1/2}$:

$$(\mathbf{W}^{1/2} t) = (\mathbf{W}^{1/2} \Phi) w + (\mathbf{W}^{1/2} \varepsilon)$$

Define the transformed variables:

$$\tilde{t} = \mathbf{W}^{1/2} t, \qquad \tilde{\Phi} = \mathbf{W}^{1/2} \Phi, \qquad \tilde{\varepsilon} = \mathbf{W}^{1/2} \varepsilon$$

The new model is $\tilde{t} = \tilde{\Phi} w + \tilde{\varepsilon}$. Let's check the covariance of the new error $\tilde{\varepsilon}$:

$$\begin{aligned} \mathrm{Cov}(\tilde{\varepsilon}) &= \mathrm{Cov}(\mathbf{W}^{1/2} \varepsilon) \\ &= \mathbf{W}^{1/2} \mathrm{Cov}(\varepsilon) (\mathbf{W}^{1/2})^\top \\ &= \mathbf{W}^{1/2} \mathbf{\Sigma} \mathbf{W}^{1/2} \quad \text{(since } \mathbf{W} \text{ is diagonal)} \\ &= \mathbf{W}^{1/2} \mathbf{W}^{-1} \mathbf{W}^{1/2} \\ &= (\mathbf{W}^{1/2} \mathbf{W}^{-1/2})(\mathbf{W}^{-1/2} \mathbf{W}^{1/2}) = I \cdot I = I \end{aligned}$$

The transformed model has spherical errors, so we apply the OLS formula to it:

$$\hat{w} = (\tilde{\Phi}^\top \tilde{\Phi})^{-1} \tilde{\Phi}^\top \tilde{t}$$

Substitute the original variables back:

- $\tilde{\Phi}^\top \tilde{\Phi} = (\mathbf{W}^{1/2} \Phi)^\top (\mathbf{W}^{1/2} \Phi) = \Phi^\top (\mathbf{W}^{1/2})^\top \mathbf{W}^{1/2} \Phi = \Phi^\top \mathbf{W} \Phi$
- $\tilde{\Phi}^\top \tilde{t} = (\mathbf{W}^{1/2} \Phi)^\top (\mathbf{W}^{1/2} t) = \Phi^\top (\mathbf{W}^{1/2})^\top \mathbf{W}^{1/2} t = \Phi^\top \mathbf{W} t$

This yields the identical WLS estimator:

$$\hat{w}_{\mathrm{WLS}} = (\Phi^\top \mathbf{W} \Phi)^{-1} \Phi^\top \mathbf{W} t$$

4. Properties of the WLS Estimator

We assume the weights $\mathbf{W} = \mathbf{\Sigma}^{-1}$ are known.

Theorem 1.1.3 (Unbiasedness of WLS)**.** *The WLS estimator is unbiased.*

Proof. Substitute $t = \Phi w + \varepsilon$ into the estimator:

$$\begin{aligned} \hat{w}_{\mathrm{WLS}} &= (\Phi^\top \mathbf{W} \Phi)^{-1} \Phi^\top \mathbf{W} (\Phi w + \varepsilon) \\ &= (\Phi^\top \mathbf{W} \Phi)^{-1} (\Phi^\top \mathbf{W} \Phi) w + (\Phi^\top \mathbf{W} \Phi)^{-1} \Phi^\top \mathbf{W} \varepsilon \\ &= w + (\Phi^\top \mathbf{W} \Phi)^{-1} \Phi^\top \mathbf{W} \varepsilon \end{aligned}$$

Now take the expectation:

$$\begin{aligned} \mathbb{E}[\hat{w}_{\mathrm{WLS}}] &= \mathbb{E}[w] + \mathbb{E}[(\Phi^\top \mathbf{W} \Phi)^{-1} \Phi^\top \mathbf{W} \varepsilon] \\ &= w + (\Phi^\top \mathbf{W} \Phi)^{-1} \Phi^\top \mathbf{W} \mathbb{E}[\varepsilon] \\ &= w + 0 = w \end{aligned}$$

□

Theorem 1.1.4 (Covariance of WLS)**.** *The covariance matrix of the WLS estimator is* $\mathrm{Cov}(\hat{w}_{\mathrm{WLS}}) = (\Phi^\top \mathbf{W} \Phi)^{-1}$.

Proof. Using the result from the unbiasedness proof:

$$\hat{w}_{\text{WLS}} - w = (\Phi^\top \mathbf{W}\Phi)^{-1}\Phi^\top \mathbf{W}\varepsilon$$

Let $A = (\Phi^\top \mathbf{W}\Phi)^{-1}\Phi^\top \mathbf{W}$. The covariance is:

$$\begin{aligned}
\text{Cov}(\hat{w}_{\text{WLS}}) &= \mathbb{E}[(\hat{w}_{\text{WLS}} - w)(\hat{w}_{\text{WLS}} - w)^\top] \\
&= \mathbb{E}[(A\varepsilon)(A\varepsilon)^\top] = \mathbb{E}[A\varepsilon\varepsilon^\top A^\top] \\
&= A\,\mathbb{E}[\varepsilon\varepsilon^\top]\,A^\top = A\,\boldsymbol{\Sigma}\,A^\top \\
&= \left[(\Phi^\top \mathbf{W}\Phi)^{-1}\Phi^\top \mathbf{W}\right] \cdot \boldsymbol{\Sigma} \cdot \left[(\Phi^\top \mathbf{W}\Phi)^{-1}\Phi^\top \mathbf{W}\right]^\top
\end{aligned}$$

Since $\boldsymbol{\Sigma} = \mathbf{W}^{-1}$ and $\mathbf{W}^\top = \mathbf{W}$ (it's diagonal):

$$\begin{aligned}
\text{Cov}(\hat{w}_{\text{WLS}}) &= \left[(\Phi^\top \mathbf{W}\Phi)^{-1}\Phi^\top \mathbf{W}\right]\mathbf{W}^{-1}\left[\mathbf{W}\Phi(\Phi^\top \mathbf{W}\Phi)^{-1}\right] \\
&= (\Phi^\top \mathbf{W}\Phi)^{-1}\Phi^\top(\mathbf{W}\mathbf{W}^{-1})\mathbf{W}\Phi(\Phi^\top \mathbf{W}\Phi)^{-1} \\
&= (\Phi^\top \mathbf{W}\Phi)^{-1}(\Phi^\top \mathbf{W}\Phi)(\Phi^\top \mathbf{W}\Phi)^{-1} \\
&= I \cdot (\Phi^\top \mathbf{W}\Phi)^{-1} \\
&= \boxed{(\Phi^\top \mathbf{W}\Phi)^{-1}}
\end{aligned}$$

□

By the Aitken Theorem, $\hat{w}_{\text{WLS}}$ is the BLUE for this model.

5. Feasible WLS (FWLS)

In practice, the true variances σ_i^2 (and thus $\mathbf{W}$) are unknown. **Feasible WLS** is a multi-step process to estimate them:

1. Run OLS on $t = \Phi w + \varepsilon$ to get residuals $e = t - \Phi\hat{w}_{\text{OLS}}$.
2. Model the variance. Assume σ_i^2 is a function of some variables z_i (often $z_i = \phi_i$). To ensure positivity, model $\log(e_i^2)$:
$$\log(e_i^2) = z_i^\top \gamma + \nu_i$$
3. Use this model to predict the log-variances, then get predicted variances:
$$\hat{\sigma}_i^2 = \exp(z_i^\top \hat{\gamma})$$
4. Construct the estimated weight matrix: $\hat{\mathbf{W}} = \text{diag}(1/\hat{\sigma}_1^2, \ldots, 1/\hat{\sigma}_N^2)$.
5. Run WLS using $\hat{\mathbf{W}}$:
$$\hat{w}_{\text{FWLS}} = (\Phi^\top \hat{\mathbf{W}}\Phi)^{-1}\Phi^\top \hat{\mathbf{W}} t$$

This estimator is consistent and asymptotically more efficient than OLS.

Derivation of WLS as an MLE

We can derive the WLS estimator by finding the Maximum Likelihood Estimate (MLE) for a linear model with independent, heteroscedastic Gaussian noise.

1. Probabilistic Model & Error Function Assume the linear model $t_i = \phi_i^\top \mathbf{w} + \epsilon_i$, where the noise ϵ_i for each observation is drawn from a Gaussian with its own variance σ_i^2:

$$\epsilon_i \sim \mathcal{N}(0, \sigma_i^2)$$

The likelihood for a single observation t_i is:

$$p(t_i \mid \phi_i, \mathbf{w}, \sigma_i^2) = \mathcal{N}(t_i \mid \phi_i^\top \mathbf{w}, \sigma_i^2) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(t_i - \phi_i^\top \mathbf{w})^2}{2\sigma_i^2}\right)$$

The log-likelihood for the entire dataset (assuming independence) is the sum:

$$\mathcal{L}(\mathbf{w}) = \log \prod_{i=1}^{N} p(t_i) = \sum_{i=1}^{N} \log p(t_i)$$

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^{N} \left[\log\left(\frac{1}{\sqrt{2\pi\sigma_i^2}}\right) - \frac{1}{2\sigma_i^2}(t_i - \phi_i^\top \mathbf{w})^2\right]$$

To find the MLE for $\mathbf{w}$, we maximize $\mathcal{L}(\mathbf{w})$. The first term in the sum is constant w.r.t. $\mathbf{w}$, so maximizing the log-likelihood is equivalent to minimizing the negative of the second term:

$$\hat{\mathbf{w}}_{\text{MLE}} = \arg\min_{\mathbf{w}} \sum_{i=1}^{N} \frac{1}{2\sigma_i^2}(t_i - \phi_i^\top \mathbf{w})^2$$

Dropping the constant $1/2$ and defining the **weights** as $w_i = 1/\sigma_i^2$, we get the WLS error function $E(\mathbf{w})$:

$$E(\mathbf{w}) = \sum_{i=1}^{N} w_i(t_i - \phi_i^\top \mathbf{w})^2$$

2. Error Function in Matrix Form Let $\mathbf{t}$ be the $N \times 1$ target vector, $\boldsymbol{\Phi}$ the $N \times D$ design matrix, and $\mathbf{W}$ the $N \times N$ diagonal matrix of weights:

$$\mathbf{W} = \text{diag}(w_1, w_2, \ldots, w_N)$$

The error function $E(\mathbf{w})$ in matrix form is the quadratic form:

$$E(\mathbf{w}) = (\mathbf{t} - \boldsymbol{\Phi}\mathbf{w})^\top \mathbf{W}(\mathbf{t} - \boldsymbol{\Phi}\mathbf{w})$$

3. Derivation of the Closed-Form Solution To find the $\mathbf{w}$ that minimizes $E(\mathbf{w})$, we expand the expression and compute the gradient.

$$E(\mathbf{w}) = (\mathbf{t}^\top - \mathbf{w}^\top \boldsymbol{\Phi}^\top)\mathbf{W}(\mathbf{t} - \boldsymbol{\Phi}\mathbf{w})$$

$$E(\mathbf{w}) = \mathbf{t}^\top \mathbf{W}\mathbf{t} - \mathbf{t}^\top \mathbf{W}\boldsymbol{\Phi}\mathbf{w} - \mathbf{w}^\top \boldsymbol{\Phi}^\top \mathbf{W}\mathbf{t} + \mathbf{w}^\top \boldsymbol{\Phi}^\top \mathbf{W}\boldsymbol{\Phi}\mathbf{w}$$

Since $\mathbf{W}$ is symmetric ($\mathbf{W}^\top = \mathbf{W}$), the two middle terms are transposes of each other (and are scalars), so we can combine them:

$$E(\mathbf{w}) = \mathbf{t}^\top \mathbf{W}\mathbf{t} - 2\mathbf{w}^\top \boldsymbol{\Phi}^\top \mathbf{W}\mathbf{t} + \mathbf{w}^\top(\boldsymbol{\Phi}^\top \mathbf{W}\boldsymbol{\Phi})\mathbf{w}$$

Now, we take the gradient with respect to $\mathbf{w}$:

$$\nabla_{\mathbf{w}} E(\mathbf{w}) = -2\boldsymbol{\Phi}^\top \mathbf{W}\mathbf{t} + 2(\boldsymbol{\Phi}^\top \mathbf{W}\boldsymbol{\Phi})\mathbf{w}$$

Set the gradient to zero to find the minimum:

$$\mathbf{0} = -2\boldsymbol{\Phi}^\top \mathbf{W}\mathbf{t} + 2(\boldsymbol{\Phi}^\top \mathbf{W}\boldsymbol{\Phi})\mathbf{w}$$

$$(\boldsymbol{\Phi}^\top \mathbf{W}\boldsymbol{\Phi})\mathbf{w} = \boldsymbol{\Phi}^\top \mathbf{W}\mathbf{t}$$

Assuming the matrix $(\boldsymbol{\Phi}^\top \mathbf{W}\boldsymbol{\Phi})$ is invertible, we solve for $\mathbf{w}$ to get the WLS solution:

$$\boxed{\hat{\mathbf{w}}_{\text{WLS}} = (\boldsymbol{\Phi}^\top \mathbf{W}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top \mathbf{W}\mathbf{t}}$$

Effects of Data Transformations on OLS Solution

We analyze the effect of common data operations on the OLS closed-form solution $\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$.

Scaling Individual Features

- **Operation:** Scale a single feature column j (assuming $j \geq 1$, not the bias) by a constant c.
- **Proof:** Let $\mathbf{C}$ be a diagonal matrix with $\mathbf{C}_{jj} = c$ and all other diagonal entries as 1. The new design matrix is $\mathbf{X}' = \mathbf{X}\mathbf{C}$.

$$\begin{aligned}
\hat{\mathbf{w}}' &= ((\mathbf{X}\mathbf{C})^T(\mathbf{X}\mathbf{C}))^{-1}(\mathbf{X}\mathbf{C})^T\mathbf{y} \\
&= (\mathbf{C}^T\mathbf{X}^T\mathbf{X}\mathbf{C})^{-1}\mathbf{C}^T\mathbf{X}^T\mathbf{y} \\
&= \mathbf{C}^{-1}(\mathbf{X}^T\mathbf{X})^{-1}(\mathbf{C}^T)^{-1}\mathbf{C}^T\mathbf{X}^T\mathbf{y} \\
&= \mathbf{C}^{-1}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \mathbf{C}^{-1}\hat{\mathbf{w}}
\end{aligned}$$

- **Effect:** The new weight vector is $\hat{\mathbf{w}}' = \mathbf{C}^{-1}\hat{\mathbf{w}}$. Since $\mathbf{C}^{-1}$ is a diagonal matrix with $(\mathbf{C}^{-1})_{jj} = 1/c$, the corresponding weight $\hat{w}_j$ is scaled by $1/c$ ($\hat{w}'_j = \hat{w}_j/c$). All other weights, including the bias term $\hat{w}_0$, are unchanged.

Scaling All Features (not bias)

- **Operation:** Scale all feature columns $\mathbf{x}_j$ (for $j \geq 1$) by a constant c.
- **Proof:** This is the same as above, but $\mathbf{C} = \mathrm{diag}(1, c, c, \ldots, c)$. The inverse is $\mathbf{C}^{-1} = \mathrm{diag}(1, 1/c, 1/c, \ldots, 1/c)$. The proof $\hat{\mathbf{w}}' = \mathbf{C}^{-1}\hat{\mathbf{w}}$ is identical.
- **Effect:** The bias (intercept) term $\hat{w}_0$ is unchanged. All other feature weights $\hat{w}_j$ (for $j \geq 1$) are scaled by $1/c$.

Scaling Labels

- **Operation:** Scale the target vector $\mathbf{y}$ by a constant c. $\mathbf{y}' = c\mathbf{y}$.
- **Proof:**

$$\begin{aligned}
\hat{\mathbf{w}}' &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}' \\
&= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T(c\mathbf{y}) \\
&= c \cdot [(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}] = c \cdot \hat{\mathbf{w}}
\end{aligned}$$

- **Effect:** All weights, including the bias term, are scaled by c.

Duplicating Rows

- **Operation:** Stack the entire dataset $(\mathbf{X}, \mathbf{y})$ on top of itself.
- **Proof:** The new matrices are $\mathbf{X}' = \begin{pmatrix} \mathbf{X} \\ \mathbf{X} \end{pmatrix}$ and $\mathbf{y}' = \begin{pmatrix} \mathbf{y} \\ \mathbf{y} \end{pmatrix}$.

$$\begin{aligned}
(\mathbf{X}')^T\mathbf{X}' &= \begin{pmatrix} \mathbf{X}^T & \mathbf{X}^T \end{pmatrix} \begin{pmatrix} \mathbf{X} \\ \mathbf{X} \end{pmatrix} = 2(\mathbf{X}^T\mathbf{X}) \\
(\mathbf{X}')^T\mathbf{y}' &= \begin{pmatrix} \mathbf{X}^T & \mathbf{X}^T \end{pmatrix} \begin{pmatrix} \mathbf{y} \\ \mathbf{y} \end{pmatrix} = 2(\mathbf{X}^T\mathbf{y}) \\
\hat{\mathbf{w}}' &= (2(\mathbf{X}^T\mathbf{X}))^{-1}(2(\mathbf{X}^T\mathbf{y})) \\
&= \frac{1}{2}(\mathbf{X}^T\mathbf{X})^{-1}(2\mathbf{X}^T\mathbf{y}) = \hat{\mathbf{w}}
\end{aligned}$$

- **Effect:** The solution $\hat{\mathbf{w}}$ is unchanged.

Removing Bias Term

- **Operation:** The original matrix $\mathbf{X} = [\mathbf{1}, \mathbf{X}_f]$ (where $\mathbf{X}_f$ are the features) becomes $\mathbf{X}' = \mathbf{X}_f$.
- **Proof:** The new solution $\hat{\mathbf{w}}' = (\mathbf{X}_f^T\mathbf{X}_f)^{-1}\mathbf{X}_f^T\mathbf{y}$ is not trivially related to the original $\hat{\mathbf{w}}$.
- **Effect:** The solution changes completely. The new model is forced to pass through the origin, which alters all coefficients.

Adding Dummy/Constant Features

- **Operation:** Add a new feature column that is constant, e.g., $\mathbf{x}_{\text{new}} = c \cdot \mathbf{1}$.
- **Proof:** The original matrix $\mathbf{X}$ already has a bias column (a column of 1s). The new column is a perfect linear combination of the bias column ($\mathbf{x}_{\text{new}} = c \cdot \mathbf{x}_0$). This is **perfect multicollinearity**.
- **Effect:** The columns of $\mathbf{X}'$ are linearly dependent, so the Gram matrix $(\mathbf{X}')^T\mathbf{X}'$ is singular (not invertible). A unique closed-form solution does not exist.

Duplicating Features

- **Operation:** Add a new feature column $\mathbf{x}_k$ that is identical to an existing column $\mathbf{x}_j$.
- **Proof:** The new column $\mathbf{x}_k$ is a perfect linear combination of $\mathbf{x}_j$ (i.e., $\mathbf{x}_k = 1 \cdot \mathbf{x}_j$). This is **perfect multicollinearity**.
- **Effect:** The Gram matrix $(\mathbf{X}')^T\mathbf{X}'$ is singular. A unique closed-form solution does not exist.

Adding a Single Data Row

- **Operation:** Add a new row $[\mathbf{x}_{\text{new}}^\top, y_{\text{new}}]$ to the dataset.
- **Proof:** The new matrices are $\mathbf{X}' = \begin{pmatrix} \mathbf{X} \\ \mathbf{x}_{\text{new}}^\top \end{pmatrix}$ and $\mathbf{y}' = \begin{pmatrix} \mathbf{y} \\ y_{\text{new}} \end{pmatrix}$.

$$\begin{aligned}
(\mathbf{X}')^T\mathbf{X}' &= (\mathbf{X}^T\mathbf{X} + \mathbf{x}_{\text{new}}\mathbf{x}_{\text{new}}^\top) \\
(\mathbf{X}')^T\mathbf{y}' &= (\mathbf{X}^T\mathbf{y} + \mathbf{x}_{\text{new}}y_{\text{new}}) \\
\hat{\mathbf{w}}' &= (\mathbf{X}^T\mathbf{X} + \mathbf{x}_{\text{new}}\mathbf{x}_{\text{new}}^\top)^{-1}(\mathbf{X}^T\mathbf{y} + \mathbf{x}_{\text{new}}y_{\text{new}})
\end{aligned}$$

- **Effect:** The solution $\hat{\mathbf{w}}$ changes. The new solution can be found from the old one using the Sherman-Morrison formula for rank-1 updates, but it is not a simple scaling.

Chapter 2

Linear Discriminative Models

Quadratic Discriminant Analysis

1. Model assumptions

We consider a K-class classification problem. For each class $k \in \{1, \dots, K\}$ assume the class-conditional density is multivariate normal:

$$x \mid y = k \sim \mathcal{N}(\mu_k, \Sigma_k),$$

with class-specific mean $\mu_k \in \mathbb{R}^d$ and covariance matrix $\Sigma_k \in \mathbb{R}^{d \times d}$ (symmetric, positive definite). Class priors are

$$\pi_k = P(y = k), \qquad \sum_{k=1}^{K} \pi_k = 1.$$

A new point x is classified by choosing the class with largest posterior $p(y = k \mid x)$ (MAP rule).

2. Likelihood $p(x \mid y = k)$

The Gaussian density for class k is

$$p(x \mid y = k) = \frac{1}{(2\pi)^{d/2} |\Sigma_k|^{1/2}} \exp\Big(-\tfrac{1}{2}(x - \mu_k)^\top \Sigma_k^{-1} (x - \mu_k)\Big).$$

3. Posterior via Bayes' rule and the decision rule

By Bayes' rule the posterior is

$$p(y = k \mid x) = \frac{p(x \mid y = k)\, \pi_k}{\sum_{j=1}^{K} p(x \mid y = j)\, \pi_j}.$$

The MAP classifier chooses

$$\hat{y}(x) = \arg\max_k \; p(y = k \mid x) = \arg\max_k \; \big[\log p(x \mid y = k) + \log \pi_k\big].$$

Define the discriminant function

$$g_k(x) = \log p(x \mid y = k) + \log \pi_k.$$

Substituting the Gaussian density gives

$$g_k(x) = -\frac{d}{2} \log(2\pi) - \frac{1}{2} \log |\Sigma_k| - \frac{1}{2}(x - \mu_k)^\top \Sigma_k^{-1} (x - \mu_k) + \log \pi_k.$$

Dropping the common constant $-\frac{d}{2}\log(2\pi)$ yields the equivalent discriminant

$$\tilde{g}_k(x) = -\tfrac{1}{2}(x - \mu_k)^\top \Sigma_k^{-1} (x - \mu_k) - \tfrac{1}{2} \log |\Sigma_k| + \log \pi_k,$$

which is a quadratic function of x and thus induces quadratic decision boundaries in general.

4. Pairwise decision boundary (between classes i and j)

Class i is preferred to class j when $g_i(x) > g_j(x)$. The boundary $g_i(x) = g_j(x)$ is given by

$$-\tfrac{1}{2}(x - \mu_i)^\top \Sigma_i^{-1} (x - \mu_i) - \tfrac{1}{2} \log |\Sigma_i| + \log \pi_i = -\tfrac{1}{2}(x - \mu_j)^\top \Sigma_j^{-1} (x - \mu_j) - \tfrac{1}{2} \log |\Sigma_j| + \log \pi_j.$$

Bringing all terms to one side and multiplying by -2 yields

$$(x - \mu_i)^\top \Sigma_i^{-1} (x - \mu_i) - (x - \mu_j)^\top \Sigma_j^{-1} (x - \mu_j) + \log \frac{|\Sigma_i|}{|\Sigma_j|} - 2 \log \frac{\pi_i}{\pi_j} = 0.$$

Expanding the quadratic forms gives the standard quadratic form

$$x^\top (\Sigma_i^{-1} - \Sigma_j^{-1}) x - 2(\mu_i^\top \Sigma_i^{-1} - \mu_j^\top \Sigma_j^{-1}) x + (\mu_i^\top \Sigma_i^{-1} \mu_i - \mu_j^\top \Sigma_j^{-1} \mu_j) + \log \frac{|\Sigma_i|}{|\Sigma_j|} - 2 \log \frac{\pi_i}{\pi_j} = 0.$$

Define

$$\begin{aligned} A_{ij} &= \tfrac{1}{2}(\Sigma_i^{-1} - \Sigma_j^{-1}) \quad \text{(symmetric)}, \\ b_{ij} &= \Sigma_i^{-1} \mu_i - \Sigma_j^{-1} \mu_j, \\ c_{ij} &= -\tfrac{1}{2}(\mu_i^\top \Sigma_i^{-1} \mu_i - \mu_j^\top \Sigma_j^{-1} \mu_j) - \tfrac{1}{2} \log \frac{|\Sigma_i|}{|\Sigma_j|} + \log \frac{\pi_i}{\pi_j}. \end{aligned}$$

The boundary can be written compactly as

$$x^\top A_{ij} x + b_{ij}^\top x + c_{ij} = 0.$$

If $A_{ij} \neq 0$ (i.e. $\Sigma_i \neq \Sigma_j$) the boundary is truly quadratic (ellipses, hyperbolas, etc. depending on the signature of A_{ij}).

5. Special case: Linear Discriminant Analysis (LDA)

If all classes share a common covariance $\Sigma_k = \Sigma$ for all k, then $\Sigma_i^{-1} - \Sigma_j^{-1} = 0$ and $A_{ij} = 0$, so the quadratic terms cancel. The discriminant simplifies to a linear function:

$$\tilde{g}_k(x) = x^\top \Sigma^{-1}\mu_k - \tfrac{1}{2}\mu_k^\top \Sigma^{-1}\mu_k + \log \pi_k.$$

The pairwise boundary between classes i, j reduces to

$$(\Sigma^{-1}(\mu_i - \mu_j))^\top x + \Big(-\tfrac{1}{2}(\mu_i^\top \Sigma^{-1}\mu_i - \mu_j^\top \Sigma^{-1}\mu_j) + \log \tfrac{\pi_i}{\pi_j}\Big) = 0,$$

which is linear in x. This is the LDA decision rule.

6. MLE parameter estimates (supervised)

Given labeled training data $\{(x_n, y_n)\}_{n=1}^N$, let $N_k = \sum_{n=1}^N \mathbf{1}\{y_n = k\}$ and denote class-k samples by $x_n^{(k)}$. The log-likelihood of the labeled data is

$$L(\{\mu_k, \Sigma_k, \pi_k\}) = \sum_{n=1}^{N} \log\big(\pi_{y_n} \mathcal{N}(x_n \mid \mu_{y_n}, \Sigma_{y_n})\big).$$

Maximizing this yields closed-form MLEs:

$$\hat{\pi}_k = \frac{N_k}{N}, \qquad \hat{\mu}_k = \frac{1}{N_k} \sum_{n: y_n = k} x_n,$$

and the class-conditional covariance (MLE convention)

$$\hat{\Sigma}_k = \frac{1}{N_k} \sum_{n: y_n = k} (x_n - \hat{\mu}_k)(x_n - \hat{\mu}_k)^\top.$$

(An unbiased sample covariance uses denominator $N_k - 1$ instead of N_k; the MLE uses N_k.)

Linear Discriminant Analysis (LDA)

1. Model assumptions

We consider a K-class classification problem and assume for each class k:

$$x \mid y = k \sim \mathcal{N}(\mu_k, \Sigma),$$

i.e. all classes share the same covariance matrix $\Sigma \in \mathbb{R}^{d\times d}$ (symmetric, positive definite) but have class-specific means $\mu_k \in \mathbb{R}^d$. Class priors are

$$\pi_k = P(y = k), \qquad \sum_{k=1}^{K} \pi_k = 1.$$

2. Class-conditional likelihood

For class k the density is

$$p(x \mid y = k) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\Big(-\tfrac{1}{2}(x - \mu_k)^\top \Sigma^{-1}(x - \mu_k)\Big).$$

3. Posterior via Bayes' rule

By Bayes' rule,

$$p(y = k \mid x) = \frac{\pi_k\, p(x \mid y = k)}{\sum_{j=1}^K \pi_j\, p(x \mid y = j)}.$$

The MAP classifier chooses

$$\hat{y}(x) = \arg\max_k\ p(y = k \mid x) = \arg\max_k\ \big[\log p(x \mid y = k) + \log \pi_k\big].$$

Define the discriminant

$$g_k(x) = \log p(x \mid y = k) + \log \pi_k.$$

4. Deriving the discriminant function

Substituting the Gaussian likelihood gives

$$g_k(x) = -\tfrac{d}{2}\log(2\pi) - \tfrac{1}{2}\log|\Sigma| - \tfrac{1}{2}(x - \mu_k)^\top \Sigma^{-1}(x - \mu_k) + \log \pi_k.$$

Dropping the class-independent terms $-\frac{d}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma|$ yields an equivalent discriminant

$$\tilde{g}_k(x) = -\tfrac{1}{2}(x - \mu_k)^\top \Sigma^{-1}(x - \mu_k) + \log \pi_k.$$

Expand the quadratic form:

$$(x - \mu_k)^\top \Sigma^{-1}(x - \mu_k) = x^\top \Sigma^{-1} x - 2\mu_k^\top \Sigma^{-1} x + \mu_k^\top \Sigma^{-1}\mu_k.$$

Since $-\frac{1}{2}x^\top \Sigma^{-1} x$ is common to all classes, it can be dropped, leaving the affine discriminant

$$g_k(x) = \mu_k^\top \Sigma^{-1} x - \tfrac{1}{2}\mu_k^\top \Sigma^{-1}\mu_k + \log \pi_k,$$

which is linear in x (affine decision function).

5. Pairwise class boundary

The boundary between classes i and j is given by $g_i(x) = g_j(x)$, i.e.

$$\mu_i^\top \Sigma^{-1} x - \tfrac{1}{2}\mu_i^\top \Sigma^{-1}\mu_i + \log \pi_i = \mu_j^\top \Sigma^{-1} x - \tfrac{1}{2}\mu_j^\top \Sigma^{-1}\mu_j + \log \pi_j.$$

Collecting terms yields a linear equation

$$(\mu_i - \mu_j)^\top \Sigma^{-1} x = \tfrac{1}{2}\big(\mu_i^\top \Sigma^{-1}\mu_i - \mu_j^\top \Sigma^{-1}\mu_j\big) + \log \frac{\pi_i}{\pi_j}.$$

Equivalently,

$$w_{ij}^\top x + b_{ij} = 0, \quad \text{with} \quad w_{ij} = \Sigma^{-1}(\mu_i - \mu_j), \qquad b_{ij} = -\tfrac{1}{2}(\mu_i^\top \Sigma^{-1}\mu_i - \mu_j^\top \Sigma^{-1}\mu_j) + \log \frac{\pi_i}{\pi_j}.$$

Thus LDA yields hyperplane decision boundaries.

6. MLE parameter estimation

Given labeled data $\{(x_n, y_n)\}_{n=1}^N$, let $N_k = \sum_{n=1}^N \mathbf{1}\{y_n = k\}$ and denote class-k samples by $x_n^{(k)}$. The MLEs are

$$\hat{\pi}_k = \frac{N_k}{N}, \qquad \hat{\mu}_k = \frac{1}{N_k} \sum_{n: y_n = k} x_n.$$

The pooled (shared) covariance MLE is

$$\hat{\Sigma} = \frac{1}{N} \sum_{k=1}^{K} \sum_{n: y_n = k} (x_n - \hat{\mu}_k)(x_n - \hat{\mu}_k)^\top.$$

(Note: the MLE uses denominator N; an unbiased pooled estimator uses denominator $N - K$.)

7. Geometry of LDA decision boundaries

- With equal covariance the discriminants differ only by a linear term $\mu_k^\top \Sigma^{-1} x$ and a bias $-\frac{1}{2}\mu_k^\top \Sigma^{-1}\mu_k + \log \pi_k$. Pairwise boundaries are parallel hyperplanes determined by $w_{ij} = \Sigma^{-1}(\mu_i - \mu_j)$.
- If priors are equal ($\pi_i = \pi_j$), classification reduces to choosing the class with smallest Mahalanobis distance $\|x - \mu_k\|_{\Sigma^{-1}} = \sqrt{(x-\mu_k)^\top \Sigma^{-1}(x-\mu_k)}$.
- The orientation of the boundary depends on Σ^{-1} and the difference of means: $w_{ij} = \Sigma^{-1}(\mu_i - \mu_j)$.

Gaussian Naive Bayes

1. Assumptions

We consider a K-class problem with feature vector $\mathbf{x} = (x_1, \dots, x_d)^\top \in \mathbb{R}^d$. Naive Bayes assumes conditional feature independence given the class:

$$p(\mathbf{x} \mid y = k) = \prod_{j=1}^{d} p(x_j \mid y = k).$$

Under Gaussian feature models each conditional distribution is univariate Gaussian:

$$x_j \mid (y = k) \sim \mathcal{N}(\mu_{kj}, \sigma_{kj}^2),$$

with class- and feature-specific parameters μ_{kj} and σ_{kj}^2. Class priors are

$$\pi_k = P(y = k), \qquad \sum_{k=1}^{K} \pi_k = 1.$$

2. Class-conditional likelihood

By independence and Gaussianity,

$$p(\mathbf{x} \mid y = k) = \prod_{j=1}^{d} \frac{1}{\sqrt{2\pi}\,\sigma_{kj}} \exp\Big(-\frac{(x_j - \mu_{kj})^2}{2\sigma_{kj}^2}\Big)$$

$$= (2\pi)^{-d/2} \Big(\prod_{j=1}^{d} \sigma_{kj}\Big)^{-1} \exp\Big(-\tfrac{1}{2}\sum_{j=1}^{d} \frac{(x_j - \mu_{kj})^2}{\sigma_{kj}^2}\Big).$$

3. Posterior and discriminant function

By Bayes' rule,

$$p(y = k \mid \mathbf{x}) = \frac{\pi_k\, p(\mathbf{x} \mid y = k)}{\sum_{r=1}^{K} \pi_r\, p(\mathbf{x} \mid y = r)}.$$

For classification it suffices to compare the (log) unnormalized posterior:

$$g_k(\mathbf{x}) = \log \pi_k + \log p(\mathbf{x} \mid y = k).$$

Dropping the constant $-\frac{d}{2}\log(2\pi)$ yields

$$g_k(\mathbf{x}) = \log \pi_k - \tfrac{1}{2}\sum_{j=1}^{d} \log \sigma_{kj}^2 - \tfrac{1}{2}\sum_{j=1}^{d} \frac{(x_j - \mu_{kj})^2}{\sigma_{kj}^2}.$$

The classifier predicts

$$\hat{y}(\mathbf{x}) = \arg\max_k \; g_k(\mathbf{x}).$$

Note that $g_k(\mathbf{x})$ is additive across features, enabling per-feature computations.

4. Pairwise boundary (binary $K = 2$)

For classes 1 and 2 the decision surface $g_1(\mathbf{x}) = g_2(\mathbf{x})$ is

$$\sum_{j=1}^{d} \left[\frac{(x_j - \mu_{2j})^2}{2\sigma_{2j}^2} - \frac{(x_j - \mu_{1j})^2}{2\sigma_{1j}^2} + \tfrac{1}{2}\log \frac{\sigma_{2j}^2}{\sigma_{1j}^2} \right] = \log \frac{\pi_1}{\pi_2}.$$

This is generally a sum of univariate quadratics in the x_j and thus a (possibly) quadratic decision surface. Special cases:

- If $\sigma_{1j}^2 = \sigma_{2j}^2 = \sigma_j^2$ for all j, the quadratic x_j^2 terms cancel and the boundary is linear:

$$\sum_{j=1}^{d} \frac{\mu_{1j} - \mu_{2j}}{\sigma_j^2}\, x_j + \text{const} = 0.$$

- If additionally $\sigma_{kj}^2 = \sigma^2$ for all k, j, the boundary reduces to a dot-product with mean difference:

$$(\mu_1 - \mu_2)^\top x + \text{const} = 0.$$

5. MLE parameter estimates (supervised training)

Given labeled data $\{(x_n, y_n)\}_{n=1}^{N}$, let $N_k = \sum_{n=1}^{N} \mathbf{1}\{y_n = k\}$.

$$\hat{\pi}_k = \frac{N_k}{N}.$$

Class- and feature-wise sample means (MLE):

$$\hat{\mu}_{kj} = \frac{1}{N_k} \sum_{n: y_n = k} x_{n,j}.$$

Class- and feature-wise sample variances (MLE convention, denominator N_k):

$$\hat{\sigma}_{kj}^2 = \frac{1}{N_k} \sum_{n: y_n = k} (x_{n,j} - \hat{\mu}_{kj})^2.$$

(An unbiased variance uses denominator $N_k - 1$; in practice enforce a variance floor $\varepsilon > 0$ to avoid division by zero.)

6. Relation to LDA / QDA

Gaussian Naive Bayes corresponds to class-specific diagonal covariance matrices:

$$\Sigma_k = \operatorname{diag}(\sigma_{k1}^2, \dots, \sigma_{kd}^2),$$

so GNB is a constrained special case of QDA. If variances are shared across classes per feature ($\sigma_{kj}^2 = \sigma_j^2$) quadratic terms cancel and boundaries become linear — resembling LDA. If the pooled shared covariance of LDA is diagonal and matches GNB's variances, LDA and Gaussian NB produce identical decision boundaries; otherwise they differ.

Chapter 3

Discriminative Models for Classification

Sigmoid Function

Definition

The logistic (sigmoid) function is defined by

$$\sigma(z) = \frac{1}{1+e^{-z}}, \qquad z \in \mathbb{R},$$

which maps real numbers to the interval $(0, 1)$.

Derivative

Differentiating σ gives

$$\sigma'(z) = \frac{d}{dz}\left(\frac{1}{1+e^{-z}}\right) = \frac{e^{-z}}{(1+e^{-z})^2}.$$

Rewriting in terms of $\sigma(z)$ yields the well-known identity

$$\boxed{\sigma'(z) = \sigma(z)\big(1-\sigma(z)\big)}.$$

Range and bounds

Since $e^{-z} > 0$ for all z, the denominator $1 + e^{-z} > 1$ and therefore

$$0 < \sigma(z) < 1 \qquad \forall z \in \mathbb{R}.$$

Symmetry

The sigmoid satisfies the symmetry relation

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

Logistic regression and log-odds

In logistic regression the class probability is modeled as

$$p(y = 1 \mid x) = \sigma(w^\top x + b).$$

The odds and log-odds are

$$\frac{p}{1-p} = e^{w^\top x + b}, \qquad \log\frac{p}{1-p} = w^\top x + b,$$

so logistic regression models the *log-odds* as an affine function of features.

Interpretation of weights

Magnitude. A unit increase in feature x_j changes the log-odds by w_j and multiplies the odds by e^{w_j}. Thus $|w_j|$ measures the strength of feature x_j.

Sign. If $w_j > 0$ increasing x_j raises $p(y = 1 \mid x)$; if $w_j < 0$ increasing x_j lowers $p(y = 1 \mid x)$; if $w_j = 0$ the feature has no effect.

Scaling weights

If $w' = \alpha w$ and $b' = \alpha b$ with $\alpha > 0$, then the decision boundary $w'^\top x + b' = 0$ is identical to $w^\top x + b = 0$ (classification unchanged), although the sigmoid becomes steeper and probabilities change. Multiplying by a negative scalar flips class labels.

Need for the bias term

Without bias $(b = 0)$:

$$p(y = 1 \mid x) = \sigma(w^\top x)$$

the decision boundary is forced through the origin $(w^\top x = 0)$ and $p(y = 1 \mid x = 0) = 0.5$. The bias term permits translation of the hyperplane and models baseline class imbalance.

Compact summary

- $\sigma(z) = \dfrac{1}{1+e^{-z}}$, range $(0, 1)$.
- $\sigma'(z) = \sigma(z)\big(1 - \sigma(z)\big)$.
- Symmetry: $\sigma(z) = 1 - \sigma(-z)$.
- Logistic regression models $\log\frac{p}{1-p} = w^\top x + b$ (log-odds linear).
- $|w_j|$ = strength; $\operatorname{sign}(w_j)$ = direction.
- Positive scaling of (w, b) leaves the decision boundary unchanged.
- Bias term is necessary to avoid forcing the boundary through the origin.

Log-Likelihood

Model

For binary logistic regression, the conditional probability of the positive class is

$$p(y = 1 \mid x; w, b) = \sigma(z), \qquad z = w^\top x + b,$$

where the sigmoid function is

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

Thus,

$$p(y = 0 \mid x; w, b) = 1 - \sigma(z),$$

and the compact expression for either label $y \in \{0, 1\}$ is

$$p(y \mid x; w, b) = \sigma(z)^{\,y}\big(1 - \sigma(z)\big)^{1-y}.$$

Likelihood of the dataset

Assuming IID samples (x_n, y_n) for $n = 1, \dots, N$, the likelihood is

$$\mathcal{L}(w, b) = \prod_{n=1}^{N} \sigma(z_n)^{y_n}\big(1 - \sigma(z_n)\big)^{1-y_n}, \qquad z_n = w^\top x_n + b.$$

Log-likelihood

Taking logs gives the standard form

$$\ell(w, b) = \sum_{n=1}^{N} \Big[y_n \log \sigma(z_n) + (1 - y_n) \log\big(1 - \sigma(z_n)\big)\Big].$$

Equivalent compact (stable) expression

Using the identity

$$y_n \log \sigma(z_n) + (1 - y_n) \log\big(1 - \sigma(z_n)\big) = y_n z_n - \log(1 + e^{z_n}),$$

the log-likelihood becomes

$$\boxed{\ell(w, b) = \sum_{n=1}^{N} \Big(y_n z_n - \log(1 + e^{z_n})\Big), \qquad z_n = w^\top x_n + b.}$$

Negative log-likelihood (binary cross-entropy)

The loss minimized in logistic regression is the negative log-likelihood:

$$\boxed{\mathcal{L}_{\text{NLL}}(w, b) = -\ell(w, b) = \sum_{n=1}^{N} \Big(\log(1 + e^{z_n}) - y_n z_n\Big).}$$

Gradient of the log-likelihood

Define $p_n = \sigma(z_n)$. Then

$$\nabla_w \ell(w, b) = \sum_{n=1}^{N} (y_n - p_n)\, x_n, \qquad \frac{\partial \ell}{\partial b} = \sum_{n=1}^{N} (y_n - p_n).$$

For the NLL (to be minimized),

$$\nabla_w \mathcal{L}_{\text{NLL}} = \sum_{n=1}^{N} (p_n - y_n) x_n, \qquad \frac{\partial \mathcal{L}_{\text{NLL}}}{\partial b} = \sum_{n=1}^{N} (p_n - y_n).$$

Hessian (optional)

Let $p_n = \sigma(z_n)$ and $R = \operatorname{diag}(p_n(1 - p_n))$. With feature matrix X (rows $x_n^\top$),

$$\nabla_w^2 \mathcal{L}_{\text{NLL}} = X^\top R X,$$

which is positive semi-definite, proving convexity of the NLL in w.

Summary

- Likelihood: $\prod_n \sigma(z_n)^{y_n}(1 - \sigma(z_n))^{1-y_n}$.
- Log-likelihood: $\ell = \sum_n (y_n z_n - \log(1 + e^{z_n}))$.
- NLL: $\mathcal{L}_{\text{NLL}} = \sum_n (\log(1 + e^{z_n}) - y_n z_n)$.
- Gradient: $\nabla_w \mathcal{L}_{\text{NLL}} = \sum_n (p_n - y_n) x_n$.

Convexity

Setup: Binary Cross-Entropy / NLL

For binary labels $y_n \in \{0, 1\}$ and data (x_n, y_n), define

$$z_n(w) = w^\top x_n + b,$$

(or equivalently treat w as augmented to include the bias).
The negative log-likelihood (binary cross-entropy) is

$$L(w) = \sum_{n=1}^{N} \Big(\log\Big(1 + e^{\,z_n(w)}\Big) - y_n z_n(w)\Big).$$

We show that $L(w)$ is convex in w.

Proof via Convex Function Composition

Define a scalar function $\phi : \mathbb{R} \to \mathbb{R}$ by

$$\phi(a) = \log(1 + e^a).$$

Compute first and second derivatives:

$$\phi'(a) = \frac{e^a}{1 + e^a} = \sigma(a),$$

$$\phi''(a) = \frac{e^a}{(1 + e^a)^2} = \sigma(a)\big(1 - \sigma(a)\big) > 0 \qquad \forall\, a \in \mathbb{R}.$$

Thus ϕ is strictly convex.
Each term in the loss is

$$\phi\big(z_n(w)\big) - y_n z_n(w).$$

Since

$$z_n(w) = w^\top x_n$$

is an affine function of w, the composition $\phi(z_n(w))$ is convex in w. The term $-y_n z_n(w)$ is affine in w and therefore convex.
A finite sum of convex functions is convex, hence

$$\boxed{L(w) \text{ is convex in } w.}$$

Matrix Form of the Log-Likelihood

Setup

Let the data matrix be

$$X \in \mathbb{R}^{N\times d}, \qquad (\text{row } n \text{ is } x_n^\top),$$

with weight vector $w \in \mathbb{R}^d$ and bias $b \in \mathbb{R}$.
Labels:

$$y \in \mathbb{R}^N, \qquad y_n \in \{0,1\}.$$

Define linear scores and probabilities:

$$z = Xw + b\mathbf{1} \in \mathbb{R}^N, \qquad p = \sigma(z) = \frac{1}{1+e^{-z}}.$$

Log-Likelihood (Vector Form)

The binary log-likelihood is

$$\ell(w,b) = \sum_{n=1}^{N} \left[y_n \log p_n + (1-y_n)\log(1-p_n)\right].$$

In vectorized form:

$$\boxed{\ell(w,b) = y^\top \log(p) \;+\; (1-y)^\top \log(1-p),}$$

where logs are applied elementwise.

Compact Form Using Linear Scores

Using

$$\log p = z - \log(1+e^z), \qquad \log(1-p) = -\log(1+e^z),$$

the log-likelihood becomes

$$\boxed{\ell(w,b) = y^\top z \;-\; \mathbf{1}^\top \log\!\big(1+e^z\big),}$$

where both exponential and logarithm act elementwise.

Augmented Representation

Define augmented feature matrix and parameter vector:

$$\tilde{X} = [X \quad \mathbf{1}], \qquad \tilde{w} = \begin{bmatrix} w \\ b \end{bmatrix}.$$

Then

$$z = \tilde{X}\tilde{w},$$

and the log-likelihood becomes

$$\boxed{\ell(\tilde{w}) = y^\top(\tilde{X}\tilde{w}) \;-\; \mathbf{1}^\top \log\!\big(1+e^{\tilde{X}\tilde{w}}\big).}$$

Log-Likelihood Limits and Perfect Separability

Sigmoid Limit Behavior

The logistic (sigmoid) function

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

satisfies

$$\lim_{z\to+\infty} \sigma(z) = 1, \qquad \lim_{z\to-\infty} \sigma(z) = 0.$$

Thus perfect classification with infinite confidence requires

$$z \to +\infty \text{ for } y=1, \qquad z \to -\infty \text{ for } y=0.$$

Log-Likelihood of a Single Sample

For binary labels,

$$\log p(y \mid x) = y\log p + (1-y)\log(1-p), \qquad p = \sigma(z).$$

Since $0 < p < 1$,

$$\log(p) \le 0, \qquad \log(1-p) \le 0,$$

hence

$$\boxed{\log p(y \mid x) \le 0.}$$

Limit for Positive Samples

For $y = 1$,

$$\log p(y=1 \mid x) = \log \sigma(z).$$

Because

$$\lim_{z\to+\infty} \sigma(z) = 1,$$

$$\lim_{z\to+\infty} \log\sigma(z) = \log(1) = 0.$$

Limit for Negative Samples

For $y = 0$,

$$\log p(y=0 \mid x) = \log(1-\sigma(z)).$$

Since

$$\lim_{z\to-\infty} \sigma(z) = 0,$$

$$\lim_{z\to-\infty} \log(1-\sigma(z)) = \log(1) = 0.$$

Implication for the Full Log-Likelihood

The full log-likelihood is

$$\ell(w) = \sum_{n=1}^{N} \log p(y_n \mid x_n).$$

To make every term tend to 0, we need

$$z_n = w^\top x_n + b \to \begin{cases} +\infty & y_n = 1, \\ -\infty & y_n = 0. \end{cases}$$

For linearly separable data, scaling $w \mapsto cw$, $c \to \infty$, yields

$$z_n(c) = c(w^\top x_n + b),$$

and therefore

$$\lim_{c\to\infty} \ell(cw) = 0.$$

Zero is the Maximum Possible Log-Likelihood

For any probability $0 < p < 1$,

$$\log p < 0.$$

Thus

$$\boxed{\ell(w) \le 0 \quad \text{for all } w,}$$

with equality only when every predicted probability equals the true label exactly.

Final Summary

$$\boxed{\ell(w) \to 0 \quad \iff \quad w^\top x_n + b \to \begin{cases} +\infty, & y_n = 1, \\ -\infty, & y_n = 0, \end{cases}}$$

This occurs only under perfect linear separability. Therefore logistic regression has **no finite maximum-likelihood solution** in separable datasets without regularization.

Gradient of Logistic Regression

Setup

Let

$$X \in \mathbb{R}^{N\times d}, \qquad y \in \mathbb{R}^N, \quad y_n \in \{0,1\}.$$

Parameters:

$$w \in \mathbb{R}^d, \qquad b \in \mathbb{R}.$$

Define logits and probabilities:

$$z = Xw + b\,\mathbf{1}, \qquad p = \sigma(z) = \frac{1}{1+e^{-z}} \in \mathbb{R}^N.$$

The negative log-likelihood (cross-entropy) is

$$\mathcal{L}(w,b) = -\sum_{n=1}^{N} \left[y_n \log p_n + (1-y_n)\log(1-p_n)\right].$$

Derivative for a Single Sample

For a single sample,

$$\ell_n = -\left[y_n \log p_n + (1-y_n)\log(1-p_n)\right].$$

Using the identity $\sigma'(z) = p(1-p)$,

$$\frac{\partial \ell_n}{\partial z_n} = p_n - y_n.$$

Gradients w.r.t. Parameters

Since

$$z_n = w^\top x_n + b, \qquad \frac{\partial z_n}{\partial w} = x_n, \qquad \frac{\partial z_n}{\partial b} = 1,$$

we obtain

$$\frac{\partial \ell_n}{\partial w} = (p_n - y_n)x_n, \qquad \frac{\partial \ell_n}{\partial b} = p_n - y_n.$$

Summing over all samples yields the full gradient.

Matrix Derivation of the Vector Form

Stacking the sample-wise derivatives:

$$p - y = \begin{bmatrix} p_1 - y_1 \\ \vdots \\ p_N - y_N \end{bmatrix} \in \mathbb{R}^N.$$

Using that

$$\nabla_w z = X, \qquad z = Xw + b\,\mathbf{1},$$

the gradient is computed as

$$\nabla_w \mathcal{L} = \sum_{n=1}^{N} (p_n - y_n)x_n = X^\top (p-y).$$

Similarly,

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_{n=1}^{N} (p_n - y_n) = \mathbf{1}^\top (p-y).$$

Final Vector and Augmented Forms

$$\boxed{\nabla_w \mathcal{L} = X^\top (p-y)}$$

$$\boxed{\frac{\partial \mathcal{L}}{\partial b} = \mathbf{1}^\top (p-y)}$$

Augment features and parameters:

$$\tilde{X} = [X \quad \mathbf{1}], \qquad \tilde{w} = \begin{bmatrix} w \\ b \end{bmatrix}.$$

Then the gradient becomes

$$\boxed{\nabla_{\tilde{w}} \mathcal{L} = \tilde{X}^\top (p-y)}$$

Gradient Descent on Logistic Regression

Setup

Given data

$$X \in \mathbb{R}^{N\times d}, \qquad y \in \mathbb{R}^N, \quad y_n \in \{0,1\},$$

and parameters

$$w \in \mathbb{R}^d, \qquad b \in \mathbb{R},$$

define logits and probabilities:

$$z = Xw + b\,\mathbf{1}, \qquad p = \sigma(z) = \frac{1}{1+e^{-z}}.$$

The negative log-likelihood (cross-entropy) is

$$\mathcal{L}(w,b) = -\sum_{n=1}^{N} \left[y_n \log p_n + (1-y_n)\log(1-p_n)\right].$$

The gradients (derived earlier) are

$$\nabla_w \mathcal{L} = X^\top (p - y), \qquad \frac{\partial \mathcal{L}}{\partial b} = \mathbf{1}^\top (p - y).$$

In augmented notation define

$$\tilde{X} = [X \quad \mathbf{1}], \qquad \tilde{w} = \begin{bmatrix} w \\ b \end{bmatrix}.$$

Then

$$\nabla_{\tilde{w}} \mathcal{L} = \tilde{X}^\top (p - y).$$

Batch Gradient Descent (BGD)

At iteration t, compute

$$g^{(t)} = \tilde{X}^\top \left(\sigma(\tilde{X} \tilde{w}^{(t)}) - y \right).$$

Update:

$$\boxed{\tilde{w}^{(t+1)} = \tilde{w}^{(t)} - \eta \, \tilde{X}^\top \Big(\sigma(\tilde{X} \tilde{w}^{(t)}) - y \Big),}$$

where $\eta > 0$ is the learning rate.
Separately for w and b,

$$w \leftarrow w - \eta \, X^\top (p - y), \qquad b \leftarrow b - \eta \, \mathbf{1}^\top (p - y).$$

L2-Regularized BGD

For objective

$$\mathcal{L}_\lambda = \mathcal{L} + \tfrac{\lambda}{2} \|w\|^2,$$

(no penalty on b),

$$\nabla_w \mathcal{L}_\lambda = X^\top (p - y) + \lambda w.$$

Thus the update becomes

$$\boxed{w \leftarrow w - \eta \big(X^\top (p - y) + \lambda w \big), \qquad b \leftarrow b - \eta \, \mathbf{1}^\top (p - y).}$$

Stochastic Gradient Descent (SGD)

Choose an index $i \in \{1, \ldots, N\}$. Compute

$$z_i = w^\top x_i + b, \qquad p_i = \sigma(z_i).$$

Per-example gradient:

$$\nabla_w \ell_i = (p_i - y_i) x_i, \qquad \frac{\partial \ell_i}{\partial b} = p_i - y_i.$$

SGD update:

$$w \leftarrow w - \eta (p_i - y_i) x_i, \qquad b \leftarrow b - \eta (p_i - y_i).$$

In augmented form let $\tilde{x}_i = \begin{bmatrix} x_i \\ 1 \end{bmatrix}$:

$$\boxed{\tilde{w} \leftarrow \tilde{w} - \eta (p_i - y_i) \tilde{x}_i.}$$

Mini-Batch SGD

For mini-batch B,

$$g_B = \sum_{n \in B} (p_n - y_n) x_n, \qquad g_{b,B} = \sum_{n \in B} (p_n - y_n).$$

Update:

$$w \leftarrow w - \eta g_B, \qquad b \leftarrow b - \eta g_{b,B}.$$

Common practice: use averaged gradients $\frac{1}{|B|} g_B$.

Regularized SGD (L2)

$$w \leftarrow w - \eta \big((p_i - y_i) x_i + \lambda w \big).$$

Final Quick-Reference Summary

$$\boxed{\text{Batch GD:} \;\; \tilde{w} \leftarrow \tilde{w} - \eta \tilde{X}^\top \big(\sigma(\tilde{X} \tilde{w}) - y \big)}$$

$$\boxed{\text{SGD (sample } i\text{)} : \;\; \tilde{w} \leftarrow \tilde{w} - \eta (\sigma(\tilde{x}_i^\top \tilde{w}) - y_i) \tilde{x}_i}$$

$$\boxed{\text{Mini-batch } B : \;\; \tilde{w} \leftarrow \tilde{w} - \eta \, \frac{1}{|B|} \, \tilde{X}_B^\top \big(\sigma(\tilde{X}_B \tilde{w}) - y_B \big)}$$

Learning Rate Stability

Logistic Loss and Gradient Descent

The logistic loss is

$$L(w) = -\sum_{n=1}^{N} \left[y_n \log p_n + (1 - y_n) \log(1 - p_n) \right], \qquad p_n = \sigma(w^\top x_n),$$

with gradient

$$\nabla L(w) = X^\top (p - y).$$

Batch gradient descent update:

$$w^{(t+1)} = w^{(t)} - \eta \, \nabla L(w^{(t)}).$$

Hessian of the Logistic Loss

The Hessian is

$$H(w) = \nabla^2 L(w) = X^\top R X, \qquad R = \operatorname{diag}\big(p_n (1 - p_n) \big).$$

Since

$$0 < p_n (1 - p_n) \le \frac{1}{4},$$

we have

$$0 \preceq R \preceq \tfrac{1}{4} I, \qquad H(w) \preceq \tfrac{1}{4} X^\top X.$$

Thus the largest eigenvalue satisfies

$$\lambda_{\max}(H(w)) \le \frac{1}{4} \lambda_{\max}(X^\top X).$$

Define the upper bound

$$L = \frac{1}{4} \lambda_{\max}(X^\top X).$$

Stability Condition for Gradient Descent

Gradient descent on a smooth convex function with Lipschitz gradient constant L is stable only if

$$0 < \eta < \frac{2}{L}.$$

Thus for logistic regression:

$$\boxed{\eta < \frac{8}{\lambda_{\max}(X^\top X)}}$$

is required for convergence.
If the condition is violated:

$$|1 - \eta\lambda_{\max}(H)| > 1 \quad \Rightarrow \quad \text{gradient descent diverges.}$$

Error Dynamics Near the Optimum

Approximate the loss near w^*:

$$L(w) \approx L(w^*) + \frac{1}{2}(w - w^*)^\top H(w^*)(w - w^*).$$

Let

$$e_t = w^{(t)} - w^*.$$

Then the update becomes

$$e_{t+1} = (I - \eta H)e_t.$$

Convergence requires

$$\rho(I - \eta H) < 1 \quad \iff \quad |1 - \eta\lambda_i(H)| < 1 \ \forall i,$$

leading to the condition

$$\boxed{0 < \eta < \frac{2}{\lambda_{\max}(H)}}.$$

If $\eta > \frac{2}{\lambda_{\max}(H)}$, then

$$|e_{t+1}| > |e_t|,$$

so parameters diverge.

Divergence Behavior Specific to Logistic Regression

If η is too large:

$$w_{t+1} = w_t - \eta X^\top(p - y)$$

becomes too large in magnitude. Then

$$|Xw| \to \infty \quad \Rightarrow \quad p = \sigma(Xw) \to 0 \text{ or } 1.$$

Since

$$-\log p \to \infty, \qquad -\log(1 - p) \to \infty,$$

the loss satisfies

$$L(w_{t+1}) \to \infty.$$

Final Summary

- Logistic loss curvature bound:

$$\lambda_{\max}(H(w)) \le \tfrac{1}{4}\lambda_{\max}(X^\top X).$$

- Stability condition:

$$0 < \eta < \frac{2}{\lambda_{\max}(H)}.$$

- Too large learning rate:

$$|w_{t+1} - w^*| > |w_t - w^*| \Rightarrow \text{divergence.}$$

- Divergence amplifies logits:

$$|Xw| \to \infty \Rightarrow p \to 0 \text{ or } 1 \Rightarrow L \to \infty.$$

Linearity of Decision Boundary

Binary Logistic Regression

The model is

$$p(y = 1 \mid x) = \sigma(w^\top x + b), \qquad \sigma(z) = \frac{1}{1 + e^{-z}}.$$

The decision rule assigns class 1 when

$$p(y = 1 \mid x) \ge 0.5.$$

Since $\sigma(z) = 0.5 \iff z = 0$, the decision boundary is obtained by solving

$$\sigma(w^\top x + b) = 0.5 \quad \iff \quad w^\top x + b = 0.$$

Thus the decision boundary is the hyperplane

$$\boxed{\{x \in \mathbb{R}^d : w^\top x + b = 0\}.}$$

Geometric Interpretation

- The normal vector of the boundary is w.
- Signed distance of x to the boundary:

$$\frac{w^\top x + b}{\|w\|}.$$

- Scaling (w, b) by any $c > 0$:

$$(cw)^\top x + cb = c(w^\top x + b) = 0,$$

  leaves the boundary unchanged.

Multiclass Extension

For softmax regression, class scores are $s_k(x) = w_k^\top x + b_k$. The boundary between classes i and j satisfies

$$s_i(x) = s_j(x) \quad \iff \quad (w_i - w_j)^\top x + (b_i - b_j) = 0,$$

which is also a hyperplane.

Summary

$$\boxed{\text{Logistic regression always yields an affine (hyperplane) decision boundary.}}$$

SVM and Logistic Crossover

Statement of the question

When do L2-regularized logistic regression and (hard- or soft-margin) SVM produce the same separating hyperplane (up to scaling)?

1. Exact equality under strong conditions

Let logistic regression solution with L2 penalty be

$$w_\lambda = \arg\min_{w,b} \sum_{n=1}^{N} \log\bigl(1 + e^{-y_n(w^\top x_n + b)}\bigr) \; + \; \tfrac{\lambda}{2}\|w\|^2.$$

Let $(w_{\mathsf{SVM}}, b_{\mathsf{SVM}})$ be the hard-margin SVM maximum-margin solution:

$$\min_{w,b} \; \tfrac{1}{2}\|w\|^2 \quad \text{s.t.} \quad y_n(w^\top x_n + b) \geq 1 \quad \forall n.$$

If the data are linearly separable and the maximum-margin direction is unique, then

$$\frac{w_\lambda}{\|w_\lambda\|} \longrightarrow \frac{w_{\mathsf{SVM}}}{\|w_{\mathsf{SVM}}\|} \qquad \text{as } \lambda \to 0^+,$$

so L2-regularized logistic regression converges to the SVM separating direction (same hyperplane up to scale).

2. Asymptotic / loss-level intuition

Define per-example margins $z = y(w^\top x + b)$. Two losses:

$$\ell_{\mathsf{log}}(z) = \log\bigl(1 + e^{-z}\bigr), \qquad \ell_{\mathsf{hinge}}(z) = \max(0, 1 - z).$$

For large positive z,

$$\ell_{\mathsf{log}}(z) \approx e^{-z} \to 0, \qquad \ell_{\mathsf{hinge}}(z) = 0.$$

When $\lambda \to 0$ (so $\|w\|$ grows), most well-separated points produce very large z and negligible contribution; only points near the margin (support-vector-like points) dominate the logistic objective. Hence the effective optimization concentrates on margin-critical points, yielding the maximum-margin direction.

3. Geometric condition (margin alignment)

Logistic and SVM align when:

1. The data are *linearly separable*.
2. The maximum-margin hyperplane is *unique* (no degenerate set of support vectors defining multiple directions).
3. Logistic regression is L2-regularized with regularization parameter λ taken sufficiently small.
4. The solution enters a regime where the sigmoid is saturated for non-critical points and only near-margin points influence the objective.

Under these geometric conditions the logistic solution approaches the SVM solution in direction.

4. When they do not coincide

The decision boundaries differ when:

- Data are not linearly separable (SVM soft-margin vs logistic probability fit).
- Logistic regularization λ is not vanishingly small.
- SVM uses a soft-margin parameter C that does not match LR's effective regularization.
- Class-conditional distributions overlap and probabilistic modelling (LR) is preferred.

5. Compact exam-ready summary

$$\boxed{\text{L2-logistic regression} \xrightarrow[\lambda \to 0^+]{} \text{maximum-margin solution} \;=\; \text{hard-margin SVM (direction-wise).}}$$

Failure of Logistic Regression

1. When does logistic regression fail (geometric view)?

Logistic regression models a linear decision surface

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

It fails geometrically in several cases.

(A) Linearly separable data (paradox)

If (w_0, b_0) satisfies

$$y_i\,(w_0^\top x_i + b_0) > 0 \quad \forall i,$$

then scaling $w \leftarrow c w_0,\; b \leftarrow c b_0$ with $c \to \infty$ yields

$$p_i = \sigma(c\, y_i(w_0^\top x_i + b_0)) \to 1, \qquad \ell(w) = \sum_i \log \sigma(c m_i) \to 0,$$

but no finite (w, b) attains $\ell = 0$.

$$\boxed{\text{Logistic regression has no finite MLE under perfect separability.}}$$

Geometric consequences: $\|w\| \to \infty$, same boundary but diverging parameters, optimization unstable without regularization.

(B) Nonlinear true boundary

If the true separator is nonlinear (circle, XOR, spiral), any affine model $w^\top x + b$ cannot fit the geometry:

$$\text{Model class too simple} \quad \Rightarrow \quad \text{systematic failure.}$$

(C) Extreme outliers

Far-away outliers can pivot the hyperplane to accommodate them, flattening the effective separator and harming generalization.

2. Collinearity / rank deficiency

If features are linearly dependent, e.g. $x_2 = \alpha x_1$, then $\operatorname{rank}(X) < d$.

Geometric implication Many weight vectors map to the same hyperplane: different w differing by a vector in $\ker(X)$ give identical predictions. Hence the decision boundary is identifiable, but the parameter vector is not.

Mathematical consequences Hessian

$$H = X^\top R X, \qquad R = \operatorname{diag}(p_n(1 - p_n))$$

satisfies $\operatorname{rank}(H) \leq \operatorname{rank}(X) < d$, so H can be singular. Consequences:

- Gradient descent still moves in $\operatorname{col}(X)$ and can converge to a solution subspace.
- Newton's method may fail (no H^{-1}).
- Multiple w achieve identical log-likelihood $\Rightarrow$ non-unique solution, large coefficient variance.

Geometric picture Features lying on the same direction allow rotating w within the nullspace without changing predictions; coefficients can blow up with canceling signs.

3. How regularization fixes these failures

Adding L_2 regularization yields a modified Hessian $H + \lambda I$ which is positive definite for $\lambda > 0$, restoring invertibility and uniqueness:

$$H + \lambda I \succ 0.$$

Regularization bounds $\|w\|$ and prevents blow-up under separability.

4. Summary

- **Separable data:** no finite MLE, $\|w\| \to \infty$ (regularize to fix).
- **Nonlinear true boundary:** model misspecification — use feature maps.
- **Outliers:** distort hyperplane, harm generalization.
- **Collinearity:** decision boundary learnable, parameters not unique; regularization restores identifiability.

L2 Regularization

1. Loss definition

Given data $X \in \mathbb{R}^{N\times d}$ (rows $x_n^\top$), labels $y \in \{0,1\}^N$, parameters $w \in \mathbb{R}^d$, bias $b \in \mathbb{R}$, define

$$z = Xw + b\mathbf{1}, \qquad p = \sigma(z).$$

The (unregularised) negative log-likelihood is

$$\mathcal{L}_{\text{NLL}}(w,b) = -\sum_{n=1}^{N} [y_n \log p_n + (1-y_n)\log(1-p_n)] = -y^\top \log p - (1-y)^\top \log(1-p).$$

Adding an L_2 penalty on the weights (not on the bias):

$$\boxed{\mathcal{L}_\lambda(w,b) = \mathcal{L}_{\text{NLL}}(w,b) + \frac{\lambda}{2}\|w\|_2^2.}$$

2. Useful identities

For each n,

$$\frac{\partial p_n}{\partial z_n} = p_n(1-p_n), \qquad \frac{\partial \mathcal{L}_{\text{NLL}}}{\partial z_n} = p_n - y_n.$$

Stacked form:

$$\nabla_z \mathcal{L}_{\text{NLL}} = p - y.$$

3. Vector gradients

Using $z = Xw + b\mathbf{1}$, the gradients of the regularised loss are:

$$\boxed{\nabla_w \mathcal{L}_\lambda = X^\top(p-y) + \lambda w}$$

$$\boxed{\frac{\partial \mathcal{L}_\lambda}{\partial b} = \mathbf{1}^\top(p-y)}$$

4. Per-sample component form

For a single example (x_n, y_n),

$$\nabla_w \ell_n = (p_n - y_n)x_n, \qquad \frac{\partial \ell_n}{\partial b} = p_n - y_n.$$

Thus,

$$\nabla_w \mathcal{L}_\lambda = \sum_{n=1}^{N}(p_n - y_n)x_n + \lambda w, \qquad \frac{\partial \mathcal{L}_\lambda}{\partial b} = \sum_{n=1}^{N}(p_n - y_n).$$

5. Augmented notation (if bias also regularised)

If we define $\tilde{X} = [X\ \mathbf{1}]$, $\tilde{w} = [w^\top\ b]^\top$, and use penalty $\frac{\lambda}{2}\|\tilde{w}\|^2$, then

$$\nabla_{\tilde{w}} \mathcal{L}_\lambda = \tilde{X}^\top(p-y) + \lambda\tilde{w}.$$

6. Final boxed results

$$\boxed{\mathcal{L}_\lambda(w,b) = -\sum_{n=1}^{N} [y_n \log p_n + (1-y_n)\log(1-p_n)] + \frac{\lambda}{2}\|w\|^2}$$

$$\boxed{\nabla_w \mathcal{L}_\lambda = X^\top(p-y) + \lambda w, \qquad \frac{\partial \mathcal{L}_\lambda}{\partial b} = \mathbf{1}^\top(p-y)}$$

L1 Regularisation and Gradient

1. Loss definition

Given $X \in \mathbb{R}^{N\times d}$ (rows $x_n^\top$), labels $y \in \{0,1\}^N$, parameters $w \in \mathbb{R}^d$ and bias $b \in \mathbb{R}$ (bias not regularized), define

$$z = Xw + b, \qquad p = \sigma(z) \quad \text{(elementwise)}.$$

The L$_1$-regularized logistic loss is

$$\boxed{\mathcal{L}_\lambda(w,b) = -\sum_{n=1}^{N}\big[y_n \log p_n + (1-y_n)\log(1-p_n)\big] + \lambda\|w\|_1,}$$

where $\|w\|_1 = \sum_{j=1}^{d}|w_j|$ and $\lambda \geq 0$.

2. Gradient of smooth part

The smooth (differentiable) part is the negative log-likelihood

$$\mathcal{L}_{\text{NLL}}(w,b) = -\sum_{n=1}^{N}\big[y_n \log p_n + (1-y_n)\log(1-p_n)\big].$$

Its gradient is

$$\nabla_w \mathcal{L}_{\text{NLL}} = X^\top(p-y), \qquad \frac{\partial \mathcal{L}_{\text{NLL}}}{\partial b} = \mathbf{1}^\top(p-y).$$

3. Subgradient of the ℓ_1 term

The subdifferential of $\|w\|_1$ is

$$\partial\|w\|_1 = \{s \in \mathbb{R}^d : \; s_j = \text{sign}(w_j) \text{ if } w_j \neq 0, \quad s_j \in [-1,1] \text{ if } w_j = 0\}.$$

Componentwise:

$$\partial_{w_j}|w_j| = \begin{cases} +1, & w_j > 0, \\ -1, & w_j < 0, \\ [-1,1], & w_j = 0. \end{cases}$$

4. Subgradient of the full objective (vector form)

By the sum rule for subgradients,

$$\partial_w \mathcal{L}_\lambda(w,b) = X^\top(p-y) + \lambda\,\partial\|w\|_1, \qquad \frac{\partial \mathcal{L}_\lambda}{\partial b} = \mathbf{1}^\top(p-y).$$

Thus any subgradient has the form

$$g = X^\top(p-y) + \lambda s, \qquad s \in \partial\|w\|_1.$$

5. Componentwise subgradient (explicit)

For each coordinate $j = 1,\dots,d$,

$$\partial_{w_j}\mathcal{L}_\lambda = x_j^\top(p-y) + \lambda\,\partial_{w_j}|w_j|,$$

i.e.

$$\partial_{w_j}\mathcal{L}_\lambda = \begin{cases} x_j^\top(p-y) + \lambda\;\text{sign}(w_j), & w_j \neq 0, \\ x_j^\top(p-y) + \lambda\,[-1,1], & w_j = 0, \end{cases}$$

where x_j denotes column j of X.

6. Optimality (KKT) conditions

A vector w^* is optimal iff $0 \in \partial_w\mathcal{L}_\lambda(w^*,b^*)$ and $\mathbf{1}^\top(p^*-y) = 0$. Componentwise:

- If $w_j^* \neq 0$: $\;x_j^\top(p^*-y) + \lambda\;\text{sign}(w_j^*) = 0$.
- If $w_j^* = 0$: $\;x_j^\top(p^*-y) \in [-\lambda,\lambda]$.

Hence coordinates with $|x_j^\top(p^*-y)| < \lambda$ must be exactly zero (sparsity condition).

7. Proximal (ISTA) update — soft-thresholding

A practical algorithm (proximal gradient / ISTA) for minimizing $\mathcal{L}_\lambda$:

1. Gradient step on smooth part:

$$u^{(t)} = w^{(t)} - \eta\,X^\top\big(\sigma(Xw^{(t)} + b) - y\big).$$

2. Proximal (soft-thresholding) step:

$$w^{(t+1)} = \mathcal{S}_{\eta\lambda}\big(u^{(t)}\big),$$

where $\mathcal{S}_\tau$ acts coordinatewise:

$$\big(\mathcal{S}_\tau(u)\big)_j = \text{sign}(u_j)\;\max\{|u_j| - \tau,\,0\}.$$

This yields sparse solutions and enforces the KKT conditions in the limit.

8. Summary (boxed)

$$\boxed{\mathcal{L}_\lambda(w,b) = -\sum_{n=1}^{N}\big[y_n\log p_n + (1-y_n)\log(1-p_n)\big] + \lambda\|w\|_1}$$

$$\boxed{\partial_w\mathcal{L}_\lambda(w,b) = X^\top(p-y) + \lambda\,\partial\|w\|_1, \qquad \frac{\partial\mathcal{L}_\lambda}{\partial b} = \mathbf{1}^\top(p-y)}$$

Componentwise optimality:

$$w_j^* \neq 0 \Rightarrow x_j^\top(p^*-y) + \lambda\,\text{sign}(w_j^*) = 0, \qquad w_j^* = 0 \Rightarrow x_j^\top(p^*-y) \in [-\lambda,\lambda].$$

Lambda Tends to Infinity

1. L2-Regularised Logistic Regression

Consider the L2-regularised logistic loss

$$\mathcal{L}_\lambda(w,b) = -\sum_{n=1}^{N}[y_n\log p_n + (1-y_n)\log(1-p_n)] + \frac{\lambda}{2}\|w\|_2^2, \qquad p_n = \sigma(w^\top x_n + b).$$

A. Behaviour of w_λ as $\lambda \to \infty$

As λ grows,

$$\mathcal{L}_\lambda(w,b) \approx \frac{\lambda}{2}\|w\|_2^2,$$

so minimisation forces

$$w_\lambda \to 0.$$

The bias b (not regularised) converges to some finite constant:

$$b_\lambda \to b_\infty \in \mathbb{R}.$$

B. Effect on linear score and probabilities

$$z = w_\lambda^\top x + b_\lambda \longrightarrow b_\infty, \qquad p(x) = \sigma(z) \to \sigma(b_\infty) = \text{constant}.$$

C. Effect on the decision boundary

The decision boundary

$$w_\lambda^\top x + b_\lambda = 0$$

collapses as $w_\lambda \to 0$. No finite separating hyperplane remains.

$$\boxed{\text{As } \lambda \to \infty, \text{ L2-logistic regression predicts a constant class (no boundary).}}$$

2. L1-Regularised Logistic Regression

The L1-regularised loss

$$\mathcal{L}_\lambda(w,b) = -\sum_{n=1}^{N}[y_n\log p_n + (1-y_n)\log(1-p_n)] + \lambda\|w\|_1$$

behaves similarly but more sharply.

A. Behaviour as $\lambda \to \infty$

The penalty $\lambda\|w\|_1$ dominates:

$$w_\lambda \to 0.$$

For sufficiently large λ, the optimal solution becomes exactly

$$w_\lambda = 0.$$

B. Effect on boundary

With $w_\lambda = 0$,

$$w_\lambda^\top x + b_\lambda = b_\lambda,$$

hence predictions become constant:

$$p(x) = \sigma(b_\lambda).$$

$$\boxed{\text{L1 with } \lambda \to \infty \ \Rightarrow\ w_\lambda = 0; \text{ decision boundary disappears.}}$$

3. Unified Result

For both L1 and L2 regularisation:

$$w_\lambda \to 0, \qquad w_\lambda^\top x + b_\lambda \to b_\lambda, \qquad p(x) = \sigma(b_\lambda) = \text{constant}.$$

Thus,

$$\boxed{\text{As } \lambda \to \infty, \text{ logistic regression collapses to a constant predictor and no decision boundary exists.}}$$

Softmax Function and Its Gradient

1. Definition

For logits $z \in \mathbb{R}^K$, the softmax function is

$$\sigma_i(z) = \frac{e^{z_i}}{\sum_{k=1}^K e^{z_k}}, \qquad i = 1, \dots, K.$$

Let $p = \sigma(z)$, so that $p_i > 0$ and $\sum_i p_i = 1$.

2. Jacobian of Softmax

Write $S = \sum_k e^{z_k}$. Then

$$\frac{\partial \sigma_i}{\partial z_j} = \frac{e^{z_i}}{S}\left(\delta_{ij} - \frac{e^{z_j}}{S}\right) = \sigma_i(\delta_{ij} - \sigma_j).$$

Hence the Jacobian matrix $J \in \mathbb{R}^{K \times K}$ is

$$\boxed{J = \operatorname{diag}(p) - p\,p^\top.}$$

3. Gradient of Cross-Entropy w.r.t. Logits

For one-hot label $y \in \{0,1\}^K$, define

$$\mathcal{L}(z) = -\sum_{i=1}^K y_i \log \sigma_i(z).$$

Using the Jacobian and the chain rule,

$$\boxed{\nabla_z \mathcal{L} = p - y.}$$

Thus the gradient w.r.t. each logit z_j is $\sigma_j - y_j$.

4. Gradient in Softmax Regression (Linear Model)

Let logits be $Z = XW + \mathbf{1}b^\top$ with:

$$X \in \mathbb{R}^{N \times d}, \quad W \in \mathbb{R}^{d \times K}, \quad b \in \mathbb{R}^K.$$

Let $Y \in \mathbb{R}^{N \times K}$ be one-hot labels and $P = \operatorname{softmax}(Z)$ applied rowwise.
The cross-entropy loss

$$\mathcal{L}(W, b) = -\sum_{n=1}^N \sum_{k=1}^K Y_{nk} \log P_{nk}$$

has gradients

$$\boxed{\nabla_W \mathcal{L} = X^\top (P - Y), \qquad \nabla_b \mathcal{L} = \mathbf{1}^\top (P - Y).}$$

5. Small Worked Example (Gradient Computation)

Consider a 3-class classifier ($K = 3$) and a single example $x \in \mathbb{R}^d$ with logits

$$z = \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}, \qquad y = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}.$$

Step 1: Compute softmax.

$$e^z = \begin{bmatrix} e^2 \\ e^1 \\ e^0 \end{bmatrix} = \begin{bmatrix} 7.389 \\ 2.718 \\ 1 \end{bmatrix}, \qquad S = 7.389 + 2.718 + 1 = 11.107.$$

$$p = \frac{e^z}{S} = \begin{bmatrix} 0.665 \\ 0.245 \\ 0.090 \end{bmatrix}.$$

Step 2: Gradient w.r.t. logits.

$$\nabla_z \mathcal{L} = p - y = \begin{bmatrix} 0.665 \\ 0.245 \\ 0.090 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \boxed{\begin{bmatrix} 0.665 \\ -0.755 \\ 0.090 \end{bmatrix}}.$$

Thus the gradient on the three logits is computed directly as $p - y$.

Step 3: Gradient w.r.t. weights (for this single example). If logits are $z = W^\top x + b$, then

$$\nabla_W \mathcal{L} = x\,(p - y)^\top.$$

Chapter 4

Gradient Descent

Descent

1. Gradient Descent as a Descent Method

Consider a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ and the GD update

$$x_{k+1} = x_k - \eta \nabla f(x_k).$$

We want $f(x_{k+1}) < f(x_k)$. This holds when the gradient is Lipschitz continuous with constant $L > 0$:

$$\boxed{\|\nabla f(x) - \nabla f(y)\| \le L\|x - y\|, \quad \forall x, y.}$$

Under this condition, the smoothness inequality states:

$$f(y) \le f(x) + \nabla f(x)^\top (y - x) + \frac{L}{2}\|y - x\|^2.$$

Setting $y = x_k - \eta \nabla f(x_k)$ gives:

$$f(x_{k+1}) \le f(x_k) - \eta \left(1 - \frac{L\eta}{2}\right) \|\nabla f(x_k)\|^2.$$

Thus we obtain the descent condition

$$\boxed{0 < \eta < \frac{2}{L} \quad \Longrightarrow \quad f(x_{k+1}) < f(x_k).}$$

2. Meaning of the Lipschitz Constant L

The Lipschitz gradient constant is

$$\boxed{L = \sup_{x \neq y} \frac{\|\nabla f(x) - \nabla f(y)\|}{\|x - y\|}.}$$

If f is twice differentiable,

$$\boxed{L = \sup_{x} \lambda_{\max}\left(\nabla^2 f(x)\right),}$$

i.e. the maximum curvature of f. Thus L bounds the Hessian:

$$\nabla^2 f(x) \preceq LI.$$

Equivalent smoothness inequality:

$$f(y) \le f(x) + \nabla f(x)^\top (y - x) + \frac{L}{2}\|y - x\|^2.$$

3. Values of L in Common ML Models

(A) Linear Regression

$$f(w) = \tfrac{1}{2}\|Xw - y\|^2, \qquad \nabla^2 f(w) = X^\top X.$$

Hence

$$\boxed{L = \lambda_{\max}(X^\top X).}$$

(B) Logistic Regression The Hessian is

$$\nabla^2 f(w) = X^\top R X, \quad R = \mathrm{diag}(p_n(1 - p_n)).$$

Since $0 < p_n(1 - p_n) \le \frac{1}{4}$,

$$X^\top R X \preceq \tfrac{1}{4} X^\top X.$$

Thus

$$\boxed{L \le \frac{1}{4}\lambda_{\max}(X^\top X).}$$

(C) Softmax Regression Same upper bound:

$$\boxed{L \le \frac{1}{4}\lambda_{\max}(X^\top X).}$$

4. Importance of L

The Lipschitz constant sets the maximum stable learning rate:

$$\boxed{0 < \eta < \frac{2}{L}.}$$

If $\eta > 2/L$, GD diverges:

$$f(x_{k+1}) > f(x_k).$$

5. Summary

$$\boxed{\nabla f \text{ is } L\text{-Lipschitz} \implies 0 < \eta < \frac{2}{L} \Rightarrow f(x_{k+1}) < f(x_k).}$$

$$\boxed{L = \sup_x \lambda_{\max}(\nabla^2 f(x)).}$$

$$\boxed{L_{\text{logistic}} \leq \tfrac{1}{4}\lambda_{\max}(X^\top X).}$$

Convergence for Quadratic Functions

Problem Setup

Consider the quadratic objective

$$f(x) = \tfrac{1}{2}x^\top Ax - b^\top x + c,$$

where $A \in \mathbb{R}^{d\times d}$ is symmetric positive definite (SPD) with

$$0 < \mu = \lambda_{\min}(A) \leq \lambda_{\max}(A) = L.$$

The unique minimiser is

$$x^\star = A^{-1}b.$$

Gradient descent with step size $\eta > 0$:

$$x_{k+1} = x_k - \eta(Ax_k - b).$$

Define the error vector $e_k := x_k - x^\star$.

Linear Error Recurrence

Compute

$$e_{k+1} = x_{k+1} - x^\star = x_k - x^\star - \eta A(x_k - x^\star) = (I - \eta A)e_k.$$

Hence

$$e_k = (I - \eta A)^k e_0.$$

Spectral Analysis and Convergence Condition

Diagonalise $A = Q\Lambda Q^\top$ (Q orthogonal, $\Lambda = \operatorname{diag}(\lambda_i)$). Then

$$I - \eta A = Q(I - \eta\Lambda)Q^\top,$$

with eigenvalues $1 - \eta\lambda_i$.
Gradient descent converges iff the spectral radius

$$\rho(I - \eta A) = \max_i |1 - \eta\lambda_i| < 1.$$

Thus for each $\lambda \in [\mu, L]$,

$$|1 - \eta\lambda| < 1 \quad\iff\quad 0 < \eta < \frac{2}{\lambda}.$$

The most restrictive gives

$$\boxed{0 < \eta < \frac{2}{L}.}$$

Under this condition,

$$\|e_k\|_2 \leq \rho(I - \eta A)^k\,\|e_0\|_2.$$

Convergence Rate and Optimal Step Size

Define the contraction factor

$$\rho(\eta) = \max_{\lambda\in[\mu,L]} |1 - \eta\lambda| = \max\{|1 - \eta\mu|,\ |1 - \eta L|\}.$$

To minimise $\rho(\eta)$, balance endpoints:

$$1 - \eta\mu = \eta L - 1 \quad\implies\quad \eta^\star = \frac{2}{L + \mu}.$$

The optimal convergence factor is

$$\rho^\star = \frac{L - \mu}{L + \mu} = \frac{\kappa - 1}{\kappa + 1}, \qquad \kappa = \frac{L}{\mu}.$$

Thus,

$$\boxed{\|e_k\|_2 \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^k \|e_0\|_2.}$$

Convergence in Objective Value

Using

$$f(x) - f(x^\star) = \tfrac{1}{2}e^\top Ae, \qquad \mu\|e\|_2^2 \leq e^\top Ae \leq L\|e\|_2^2,$$

we obtain

$$f(x_k) - f(x^\star) \leq \tfrac{1}{2}L\,\|e_k\|_2^2 \leq \tfrac{1}{2}L\,\rho(\eta)^{2k}\,\|e_0\|_2^2.$$

Final Statement

For the SPD quadratic objective,

$$e_{k+1} = (I - \eta A)e_k, \qquad e_k = (I - \eta A)^k e_0.$$

Gradient descent converges iff

$$0 < \eta < \frac{2}{L}.$$

With optimal step size $\eta^\star = 2/(L + \mu)$,

$$\boxed{\|x_k - x^\star\|_2 \le \left(\frac{\kappa - 1}{\kappa + 1}\right)^k \|x_0 - x^\star\|_2.}$$

Convergence and Hessian

Curvature and the Hessian

For a twice-differentiable function $f : \mathbb{R}^d \to \mathbb{R}$, the curvature at a point x is described by the Hessian

$$H(x) = \nabla^2 f(x).$$

The eigenvalues of $H(x)$ quantify curvature in different directions:

$$0 \le \lambda_i(H(x)) \le L,$$

where

$$L := \max_i \lambda_i(H(x))$$

is the *maximum curvature* (Lipschitz constant of the gradient).

Why Curvature Controls the Step Size

Gradient descent performs updates

$$x_{k+1} = x_k - \eta \nabla f(x_k).$$

To ensure that the method is a descent method (i.e. $f(x_{k+1}) < f(x_k)$), the step-size must satisfy

$$\boxed{0 < \eta < \frac{2}{L}}.$$

Derivation via Smoothness Inequality

If f has an L-Lipschitz gradient, then for all x, y:

$$f(y) \le f(x) + \nabla f(x)^\top (y - x) + \frac{L}{2}\|y - x\|^2.$$

Choosing $y = x - \eta \nabla f(x)$ yields

$$f(x_{k+1}) \le f(x_k) - \eta \left(1 - \frac{L\eta}{2}\right) \|\nabla f(x_k)\|^2.$$

Descent requires

$$1 - \frac{L\eta}{2} > 0 \quad \Longleftrightarrow \quad 0 < \eta < \frac{2}{L}.$$

Thus

$$\boxed{\text{Maximum curvature } L \text{ limits the stable step-size.}}$$

Geometric Interpretation

High curvature (sharp bowl):

- Slope changes rapidly.
- Large steps overshoot.
- Small step-size required.

Low curvature (flat bowl):

- Slope changes slowly.
- Larger safe step-size.

$$\boxed{\text{Sharper curvature} \;\Rightarrow\; \text{smaller steps required.}}$$

Eigenvalue View (Quadratic Functions)

For the quadratic model

$$f(x) = \tfrac{1}{2} x^\top A x, \qquad A \succ 0,$$

gradient descent satisfies

$$e_{k+1,i} = (1 - \eta\lambda_i)\, e_{k,i}$$

in each eigen-direction of A.
Convergence requires

$$|1 - \eta\lambda_i| < 1 \quad \Longleftrightarrow \quad 0 < \eta < \frac{2}{\lambda_i}.$$

The worst-case direction uses the largest eigenvalue $L = \lambda_{\max}$:

$$\boxed{0 < \eta < \frac{2}{L}}.$$

Thus high curvature (large eigenvalue) forces smaller learning rates.

Summary

- Curvature = eigenvalues of Hessian.
- $L = \lambda_{\max}(H)$ governs smoothness.
- Gradient descent requires

$$0 < \eta < \frac{2}{L}.$$

- Larger curvature ⇒ smaller steps needed.
- Smaller curvature ⇒ larger steps possible.

$$\boxed{\textbf{Curvature directly determines the stability of gradient descent.}}$$

Chapter 5

Gradient Descent for Linear and Logistic Regression

Derivation of GD

Gradient Descent for Linear Regression

We derive gradient–descent updates for linear regression (with bias), including batch GD, SGD, and the augmented (bias-absorbed) form. We also include the L$_2$-regularized (ridge) variant.

Problem Setup (MSE Objective)

Let

$$X \in \mathbb{R}^{N\times d}, \quad y \in \mathbb{R}^N, \qquad w \in \mathbb{R}^d,\ b \in \mathbb{R}.$$

Predictions:

$$\hat{y}_n = w^\top x_n + b, \qquad r := Xw + b\mathbf{1} - y.$$

Mean–squared error (using the standard factor $1/(2N)$):

$$J(w,b) = \frac{1}{2N}\sum_{n=1}^{N}(w^\top x_n + b - y_n)^2 = \frac{1}{2N}\|Xw + b\mathbf{1} - y\|^2.$$

Gradients

Gradient w.r.t. w.

$$\nabla_w J = \frac{1}{N}X^\top(Xw + b\mathbf{1} - y) = \frac{1}{N}X^\top r.$$

Gradient w.r.t. b.

$$\partial_b J = \frac{1}{N}\mathbf{1}^\top(Xw + b\mathbf{1} - y) = \frac{1}{N}\mathbf{1}^\top r.$$

Batch Gradient Descent (Full Gradient)

$$\boxed{w \leftarrow w - \eta\,\frac{1}{N}X^\top r, \qquad b \leftarrow b - \eta\,\frac{1}{N}\mathbf{1}^\top r.}$$

Stochastic Gradient Descent (Per-example)

For a single sample (x_i, y_i), residual $r_i = w^\top x_i + b - y_i$,

$$\boxed{w \leftarrow w - \eta\, r_i\, x_i, \qquad b \leftarrow b - \eta\, r_i.}$$

For a minibatch B, replace $r_i x_i$ by $\frac{1}{|B|}\sum_{n\in B} r_n x_n$.

Augmented (Bias-Absorbed) Form

Augment features with a constant 1:

$$\tilde{X} = [\,X \quad \mathbf{1}\,] \in \mathbb{R}^{N\times(d+1)}, \qquad \tilde{w} = \begin{bmatrix} w \\ b \end{bmatrix}.$$

Objective:

$$J(\tilde{w}) = \frac{1}{2N}\|\tilde{X}\tilde{w} - y\|^2.$$

Gradient:

$$\nabla_{\tilde{w}} J = \frac{1}{N}\tilde{X}^\top(\tilde{X}\tilde{w} - y).$$

Batch GD update:

$$\boxed{\tilde{w} \leftarrow \tilde{w} - \eta\,\frac{1}{N}\tilde{X}^\top(\tilde{X}\tilde{w} - y).}$$

SGD step (single sample $\tilde{x}_i$):

$$\boxed{\tilde{w} \leftarrow \tilde{w} - \eta\, r_i\, \tilde{x}_i.}$$

L$_2$-Regularized Variant (Ridge Regression)

Add $\frac{\lambda}{2}\|w\|^2$ (bias not regularized):

$$J_\lambda(w,b) = \frac{1}{2N}\|Xw + b\mathbf{1} - y\|^2 + \frac{\lambda}{2}\|w\|^2.$$

Gradients:

$$\nabla_w J_\lambda = \frac{1}{N} X^\top r + \lambda w, \qquad \partial_b J_\lambda = \frac{1}{N} \mathbf{1}^\top r.$$

Updates:

$$\boxed{w \leftarrow w - \eta \left(\frac{1}{N} X^\top r + \lambda w \right), \qquad b \leftarrow b - \eta \, \frac{1}{N} \mathbf{1}^\top r.}$$

Numerical Notes

- Compute the residual r once per batch for efficiency.
- Batch update costs $O(Nd)$.
- For large N, use minibatches or SGD.
- Closed form solution:

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

  (or ridge variant) is exact when invertible and often faster for small/medium problems.

Gradient Descent and Convexity for Logistic Regression

Model and Loss

For data

$$X \in \mathbb{R}^{N \times d}, \qquad y \in \{0,1\}^N,$$

parameters $w \in \mathbb{R}^d$, $b \in \mathbb{R}$, define

$$z = Xw + b\mathbf{1}, \qquad p = \sigma(z), \qquad p_n = \frac{1}{1 + e^{-z_n}}.$$

Negative log-likelihood:

$$\mathcal{L}_{\mathrm{NLL}}(w,b) = -\sum_{n=1}^{N} \big[y_n \log p_n + (1 - y_n) \log(1 - p_n) \big].$$

L$_2$-regularized version (no penalty on b):

$$\mathcal{L}_\lambda(w,b) = \mathcal{L}_{\mathrm{NLL}}(w,b) + \frac{\lambda}{2} \|w\|^2.$$

Gradient (Component and Vector Forms)

Using $\sigma'(z_n) = p_n(1 - p_n)$, the per-example derivative is

$$\frac{\partial \ell_n}{\partial z_n} = p_n - y_n.$$

Stacked vector form:

$$\nabla_z \mathcal{L}_{\mathrm{NLL}} = p - y.$$

By the chain rule for $z = Xw + b\mathbf{1}$,

$$\nabla_w \mathcal{L}_{\mathrm{NLL}} = X^\top (p - y), \qquad \partial_b \mathcal{L}_{\mathrm{NLL}} = \mathbf{1}^\top (p - y).$$

With L$_2$ regularization:

$$\nabla_w \mathcal{L}_\lambda = X^\top (p - y) + \lambda w, \qquad \partial_b \mathcal{L}_\lambda = \mathbf{1}^\top (p - y).$$

Gradient Descent Updates

Let learning rate $\eta > 0$.

Unregularized GD.

$$\boxed{w^{(t+1)} = w^{(t)} - \eta \, X^\top \big(\sigma(Xw^{(t)} + b^{(t)}\mathbf{1}) - y \big)}$$

$$\boxed{b^{(t+1)} = b^{(t)} - \eta \, \mathbf{1}^\top \big(\sigma(Xw^{(t)} + b^{(t)}\mathbf{1}) - y \big)}$$

L$_2$-regularized GD.

$$\boxed{w^{(t+1)} = w^{(t)} - \eta \Big(X^\top (\sigma(Xw^{(t)} + b^{(t)}\mathbf{1}) - y) + \lambda w^{(t)} \Big)}$$

$$\boxed{b^{(t+1)} = b^{(t)} - \eta \, \mathbf{1}^\top \big(\sigma(Xw^{(t)} + b^{(t)}\mathbf{1}) - y \big)}$$

Augmented Compact Form. Let

$$\tilde{X} = [\, X \ \mathbf{1} \,], \qquad \tilde{w} = \begin{bmatrix} w \\ b \end{bmatrix}.$$

Then

$$\tilde{w}^{(t+1)} = \tilde{w}^{(t)} - \eta \, \tilde{X}^\top \big(\sigma(\tilde{X} \tilde{w}^{(t)}) - y \big).$$

Convexity via the Hessian

Let $R = \mathrm{diag}(p_n(1 - p_n)) \succeq 0$. The Hessian w.r.t. the augmented parameter $\tilde{w}$ is

$$\nabla^2_{\tilde{w}} \mathcal{L}_{\mathrm{NLL}} = \tilde{X}^\top R \tilde{X}.$$

For any $v \in \mathbb{R}^{d+1}$,

$$v^\top \tilde{X}^\top R \tilde{X} v = (\tilde{X} v)^\top R (\tilde{X} v) = \sum_{n=1}^{N} p_n (1 - p_n) (\tilde{x}_n^\top v)^2 \geq 0.$$

Hence $\mathcal{L}_{\mathrm{NLL}}$ is convex.
With L$_2$ regularization,

$$\nabla^2_w \mathcal{L}_\lambda = X^\top R X + \lambda I_d \succ 0 \quad (\lambda > 0),$$

so the loss becomes **strictly convex** and has a unique minimizer.

Strict Convexity and Degeneracies

- Without regularization, strict convexity may fail if X is rank-deficient or if some $p_n(1 - p_n) = 0$.
- With $\lambda > 0$, strict convexity is always ensured.
- For linearly separable data, the unregularized logistic loss has no finite minimizer (weights diverge).

Summary

$$\boxed{\nabla_w \mathcal{L} = X^\top(\sigma(Xw+b\mathbf{1})-y), \qquad \partial_b \mathcal{L} = \mathbf{1}^\top(\sigma(Xw+b\mathbf{1})-y)}$$

$$\boxed{\text{Hessian } = \tilde{X}^\top R \tilde{X} \succeq 0 \;\Rightarrow\; \text{convex}; \quad \lambda > 0 \Rightarrow \text{strictly convex.}}$$

Convergence Proof

Linear (Geometric) Convergence of Gradient Descent

Assumptions. Let $f : \mathbb{R}^d \to \mathbb{R}$ be continuously differentiable and satisfy

- L-smoothness (gradient L-Lipschitz):

$$\|\nabla f(x) - \nabla f(y)\| \le L\|x-y\| \quad \forall x, y \in \mathbb{R}^d,$$

  equivalently

$$f(y) \le f(x) + \nabla f(x)^\top (y-x) + \tfrac{L}{2}\|y-x\|^2 \quad \forall x, y.$$

- μ-strong convexity ($\mu > 0$):

$$f(y) \ge f(x) + \nabla f(x)^\top (y-x) + \tfrac{\mu}{2}\|y-x\|^2 \quad \forall x, y.$$

Under these assumptions f has a unique minimizer $x^\star$ with $\nabla f(x^\star) = 0$. We analyze standard gradient descent

$$x_{k+1} = x_k - \eta \nabla f(x_k),$$

with a fixed step size $\eta > 0$.

Auxiliary inequalities (standard consequences).

(A) **Smoothness $\Rightarrow$ gradient–to–suboptimality bound.** Minimizing the quadratic upper bound $u(y) = f(x) + \nabla f(x)^\top (y-x) + \frac{L}{2}\|y-x\|^2$ in y gives $y = x - \frac{1}{L}\nabla f(x)$. Plugging in and comparing with $f^\star = f(x^\star)$ yields

$$f^\star \le f(x) - \frac{1}{2L}\|\nabla f(x)\|^2,$$

hence

$$\boxed{\|\nabla f(x)\|^2 \le 2L\big(f(x) - f^\star\big).} \tag{A}$$

(B) **Strong convexity $\Rightarrow$ suboptimality controls squared distance.** Put $y = x^\star$ in the strong-convexity inequality:

$$f(x) - f^\star \ge \frac{\mu}{2}\|x - x^\star\|^2,$$

i.e.

$$\boxed{f(x) - f^\star \ge \tfrac{\mu}{2}\|x - x^\star\|^2.} \tag{B}$$

Distance contraction (main derivation). Start from the update and expand the squared error:

$$x_{k+1} - x^\star = x_k - x^\star - \eta \nabla f(x_k).$$

Thus

$$\|x_{k+1} - x^\star\|^2 = \|x_k - x^\star\|^2 - 2\eta\, \nabla f(x_k)^\top (x_k - x^\star) + \eta^2 \|\nabla f(x_k)\|^2. \tag{1}$$

Use strong convexity (apply with $x \leftarrow x_k$, $y \leftarrow x^\star$) which rearranged gives

$$\nabla f(x_k)^\top (x_k - x^\star) \ge f(x_k) - f^\star + \frac{\mu}{2}\|x_k - x^\star\|^2. \tag{2}$$

Plug (2) into (1) to obtain

$$\begin{aligned} \|x_{k+1} - x^\star\|^2 &\le \|x_k - x^\star\|^2 - 2\eta\Big(f(x_k) - f^\star + \tfrac{\mu}{2}\|x_k - x^\star\|^2\Big) + \eta^2\|\nabla f(x_k)\|^2 \\ &= (1 - \eta\mu)\|x_k - x^\star\|^2 - 2\eta\big(f(x_k) - f^\star\big) + \eta^2\|\nabla f(x_k)\|^2. \end{aligned} \tag{3}$$

Now upper-bound $\|\nabla f(x_k)\|^2$ by (A):

$$\|\nabla f(x_k)\|^2 \le 2L\big(f(x_k) - f^\star\big). \tag{4}$$

Substitute (4) into (3):

$$\|x_{k+1} - x^\star\|^2 \le (1 - \eta\mu)\|x_k - x^\star\|^2 + (-2\eta + 2\eta^2 L)\big(f(x_k) - f^\star\big). \tag{5}$$

To eliminate the suboptimality term use (B): $f(x_k) - f^\star \ge \frac{\mu}{2}\|x_k - x^\star\|^2$. Plug into (5):

$$\begin{aligned} \|x_{k+1} - x^\star\|^2 &\le (1 - \eta\mu)\|x_k - x^\star\|^2 + (-2\eta + 2\eta^2 L)\cdot \tfrac{\mu}{2}\|x_k - x^\star\|^2 \\ &= \big(1 - 2\eta\mu + \eta^2 L\mu\big)\,\|x_k - x^\star\|^2. \end{aligned} \tag{6}$$

Define

$$q(\eta) := 1 - 2\eta\mu + \eta^2 L\mu.$$

Inequality (6) is

$$\boxed{\|x_{k+1} - x^\star\|^2 \le q(\eta)\,\|x_k - x^\star\|^2}. \tag{7}$$

Choosing η and contraction factor. The quadratic $q(\eta) = 1 - 2\eta\mu + \eta^2 L\mu$ is convex in η and minimized at

$$\eta^\star = \frac{1}{L}.$$

Plug $\eta = 1/L$ into q:

$$q\Big(\tfrac{1}{L}\Big) = 1 - 2\frac{\mu}{L} + \frac{1}{L^2}L\mu = 1 - \frac{\mu}{L}.$$

Since $0 < \mu \le L$ we have $0 < 1 - \mu/L < 1$. Therefore with the canonical choice $\eta = 1/L$ we obtain the clean contraction

$$\boxed{\|x_{k+1} - x^\star\|^2 \le \Big(1 - \frac{\mu}{L}\Big)\|x_k - x^\star\|^2.} \tag{8}$$

Iterating (8) yields for all $k \ge 0$,

$$\boxed{\|x_k - x^\star\|^2 \le \Big(1 - \frac{\mu}{L}\Big)^k \|x_0 - x^\star\|^2.} \tag{9}$$

Function-value (suboptimality) decay. Use smoothness to relate distance to function value. From L-smoothness and convexity one has

$$f(x) - f^\star \le \frac{L}{2}\|x - x^\star\|^2.$$

Combining with (9),

$$f(x_k) - f^\star \le \frac{L}{2}\|x_k - x^\star\|^2 \le \frac{L}{2}\Big(1 - \frac{\mu}{L}\Big)^k \|x_0 - x^\star\|^2.$$

Hence suboptimality decays geometrically with the same base $1 - \mu/L$. Equivalently,

$$\boxed{f(x_k) - f^\star \le \frac{L}{2}\Big(1 - \frac{\mu}{L}\Big)^k \|x_0 - x^\star\|^2}.$$

(One can also derive a directly comparable bound of the form $f(x_k) - f^\star \le \big(1 - \frac{\mu}{L}\big)^k (f(x_0) - f^\star)$ by tracing function decreases; constants vary but the geometric rate $1 - \mu/L$ remains.)

Remarks and sharper constants.

- The choice $\eta = 1/L$ gives the simple contraction factor $1 - \mu/L$. This is standard and easy to remember.
- A more careful analysis (or spectral analysis for quadratic objectives) shows the optimal constant step-size $\eta_{\text{opt}} = \dfrac{2}{L + \mu}$ yields a (tighter) contraction factor for squared distance $\Big(\dfrac{L - \mu}{L + \mu}\Big)^2$, and for distance itself $\dfrac{L - \mu}{L + \mu}$. For quadratics this rate is tight.
- The ratio $\kappa := L/\mu$ (condition number) controls speed: the closer κ is to 1, the faster the convergence.

Final concise theorem statement.

Theorem 5.2.1. *Let f be L-smooth and μ-strongly convex. Run gradient descent with step size $\eta = 1/L$. Then for all $k \ge 0$,*

$$\|x_k - x^\star\|^2 \le \Big(1 - \frac{\mu}{L}\Big)^k \|x_0 - x^\star\|^2, \qquad f(x_k) - f^\star \le \frac{L}{2}\Big(1 - \frac{\mu}{L}\Big)^k \|x_0 - x^\star\|^2.$$

Thus GD converges linearly (geometrically) with contraction factor $1 - \mu/L$.

Chapter 6

Support Vector Machines

Fundamentals and geometry

Distance From a Point to a Hyperplane

Let $x_i \in \mathbb{R}^d$ be a point, and let the hyperplane be

$$H = \{x \in \mathbb{R}^d : w^\top x + b = 0\},$$

where $w \neq 0$ is the normal vector. We derive the perpendicular distance from x_i to H from first principles.

Step 1: Orthogonal projection. Let $x_i^\perp$ denote the orthogonal projection of x_i onto H. Then

$$\operatorname{dist}(x_i, H) = \|x_i - x_i^\perp\|.$$

Step 2: Parameterization of the projection. Because the difference $x_i - x_i^\perp$ must be parallel to the normal direction w, there exists a scalar $\lambda \in \mathbb{R}$ such that

$$x_i^\perp = x_i - \lambda w.$$

Step 3: Enforce that $x_i^\perp \in H$. Use the hyperplane constraint:

$$w^\top x_i^\perp + b = 0.$$

Insert the expression for $x_i^\perp$:

$$w^\top (x_i - \lambda w) + b = 0 \quad \Longrightarrow \quad w^\top x_i - \lambda\, w^\top w + b = 0.$$

Solve for λ:

$$\lambda = \frac{w^\top x_i + b}{\|w\|^2}.$$

Step 4: Compute the distance. Since $x_i - x_i^\perp = \lambda w$, the perpendicular distance is

$$\|x_i - x_i^\perp\| = |\lambda|\,\|w\| = \left|\frac{w^\top x_i + b}{\|w\|^2}\right| \|w\| = \frac{|w^\top x_i + b|}{\|w\|}.$$

Final formula.

$$\boxed{\operatorname{dist}(x_i, H) = \frac{|w^\top x_i + b|}{\|w\|}}$$

Signed distance (SVM geometric margin). For labels $y_i \in \{\pm 1\}$, the signed distance is

$$\boxed{\gamma_i = y_i \frac{w^\top x_i + b}{\|w\|}}.$$

This quantity is the *geometric margin* used in support vector machine theory.

Scale Invariance and Supporting Hyperplanes

We study how rescaling a separating hyperplane affects its functional and geometric margins, and prove that any separating hyperplane with geometric margin $\Gamma > 0$ can be rescaled so that its supporting hyperplanes become

$$w^\top x + b = +1, \qquad w^\top x + b = -1.$$

Margins and Notation

For labeled data $\{(x_i, y_i)\}_{i=1}^n$ with $y_i \in \{\pm 1\}$, consider the hyperplane

$$H := \{x \in \mathbb{R}^d : w^\top x + b = 0\}, \qquad w \neq 0.$$

Functional margin.

$$\hat{\gamma}_i(w, b) := y_i\,(w^\top x_i + b), \qquad \hat{\gamma}(w, b) := \min_i \hat{\gamma}_i(w, b).$$

Geometric margin.

$$\Gamma(w, b) := \min_i \frac{\hat{\gamma}_i(w, b)}{\|w\|} = \frac{\hat{\gamma}(w, b)}{\|w\|}.$$

Assume the data are linearly separable so that $\Gamma(w, b) > 0$.

Points at Distance Γ from the Hyperplane

The signed perpendicular distance from x to H is

$$d(x) = \frac{w^\top x + b}{\|w\|}.$$

Thus for a labeled point,

$$\gamma_i(w,b) = y_i\, d(x_i) = y_i \frac{w^\top x_i + b}{\|w\|}.$$

By definition of the geometric margin, there exists a support index s such that

$$\gamma_s(w,b) = \Gamma \quad \Longrightarrow \quad y_s(w^\top x_s + b) = \Gamma\,\|w\|.$$

Hence any point at geometric distance Γ from H satisfies

$$w^\top x + b = \pm\Gamma\,\|w\|.$$

Scale Invariance of the Hyperplane

For any nonzero scalar c, define

$$w' = cw, \qquad b' = cb.$$

The hyperplane remains unchanged:

$$\{x : w'^\top x + b' = 0\} = \{x : c(w^\top x + b) = 0\} = \{x : w^\top x + b = 0\}.$$

Under scaling,

$$\hat\gamma_i(w',b') = c\,\hat\gamma_i(w,b), \qquad \|w'\| = |c|\,\|w\|,$$

and for $c > 0$,

$$\Gamma(w',b') = \Gamma(w,b).$$

Thus scaling leaves the geometric margin invariant while rescaling all functional margins.

Normalizing to ±1

From $\hat\gamma(w,b) = \Gamma\|w\|$, choose

$$c := \frac{1}{\hat\gamma(w,b)} = \frac{1}{\Gamma\|w\|}.$$

With $w' = cw$ and $b' = cb$, every support vector x_s satisfies

$$\hat\gamma_s(w',b') = 1, \quad \Longrightarrow \quad y_s(w'^\top x_s + b') = 1.$$

Thus

$$w'^\top x_s + b' = \begin{cases} +1, & y_s = +1, \\ -1, & y_s = -1. \end{cases}$$

All points satisfy $y_i(w'^\top x_i + b') \geq 1$, and the supporting hyperplanes become

$$H'_+ := \{x : w'^\top x + b' = +1\}, \qquad H'_- := \{x : w'^\top x + b' = -1\}.$$

Relation to the Geometric Margin

The distance between two parallel affine hyperplanes $w'^\top x + b' = c_1,\ w'^\top x + b' = c_2$ is

$$\frac{|c_1 - c_2|}{\|w'\|}.$$

Thus

$$\mathrm{dist}(H'_+, H'_-) = \frac{|(+1) - (-1)|}{\|w'\|} = \frac{2}{\|w'\|}.$$

The half-margin is therefore

$$\frac{1}{\|w'\|} = \Gamma,$$

since $\|w'\| = 1/\Gamma$ under the chosen normalization.
Hence points previously at geometric distance Γ now lie exactly on the hyperplanes $w'^\top x + b' = \pm1$.

Summary

- Scaling (w,b) by any positive constant does not change the decision boundary.
- The geometric margin is invariant under such scaling.
- By choosing $c = 1/(\Gamma\|w\|)$, we enforce the minimum functional margin to equal 1.
- Under this normalization, the supporting hyperplanes become

  $$w^\top x + b = +1, \qquad w^\top x + b = -1,$$

  and they lie at geometric distance Γ from the decision boundary.

Derivation of the SVM Objective (Hard-Margin Primal)

We begin from the geometric-margin maximization problem. Dataset $\{(x_i,y_i)\}_{i=1}^n$, $y_i \in \{\pm1\}$, classifier $(w,b) \in \mathbb{R}^d \times \mathbb{R}$.

Functional margin

$$\hat\gamma(w,b) := \min_i y_i(w^\top x_i + b).$$

Geometric margin

$$\Gamma(w,b) := \frac{\hat\gamma(w,b)}{\|w\|}.$$

Original problem

$$\max_{w,b}\ \Gamma(w,b) = \max_{w,b}\ \frac{\hat\gamma(w,b)}{\|w\|}. \tag{1}$$

Normalization (remove scale invariance)

For any $c > 0$:

$$\hat\gamma(cw,cb) = c\,\hat\gamma(w,b), \qquad \|cw\| = c\|w\|,$$

hence $\Gamma(cw,cb) = \Gamma(w,b)$.
Normalize via

$$\hat\gamma(w,b) = 1. \tag{2}$$

Then

$$\max_{w,b} \frac{\hat\gamma(w,b)}{\|w\|} \Longleftrightarrow \max_{\hat\gamma(w,b)=1} \frac{1}{\|w\|}. \tag{3}$$

Inequality form

$\hat{\gamma}(w,b)=1$ is equivalent to

$$y_i(w^\top x_i+b)\ge 1, \qquad i=1,\dots,n, \tag{4}$$

with at least one active constraint.
Thus

$$\max_{w,b}\frac{1}{\|w\|} \quad \text{s.t.} \quad y_i(w^\top x_i+b)\ge 1. \tag{4'}$$

Monotone transformation of objective

Maximize $1/\|w\| \iff$ minimize $\|w\| \iff$ minimize $\frac{1}{2}\|w\|^2$.
Hence the problem becomes

$$\boxed{\min_{w,b}\ \tfrac{1}{2}\|w\|^2 \quad \text{s.t.} \quad y_i(w^\top x_i+b)\ge 1,\ i=1,\dots,n.} \tag{P}$$

Incorrect sign check

For any feasible (w,b) and any $t>1$, (tw,tb) is feasible:

$$y_i((tw)^\top x_i+tb)=t\,y_i(w^\top x_i+b)\ge t.$$

But

$$\tfrac{1}{2}\|tw\|^2=\tfrac{1}{2}t^2\|w\|^2\to\infty.$$

Therefore

$$\max\ \tfrac{1}{2}\|w\|^2 \quad \text{is unbounded and not equivalent.}$$

Final equivalence chain

$$\max_{w,b}\min_i y_i\frac{w^\top x_i+b}{\|w\|} \iff \max\frac{\hat{\gamma}(w,b)}{\|w\|} \iff \max_{\hat{\gamma}=1}\frac{1}{\|w\|}$$
$$\iff \max_{y_i(w^\top x_i+b)\ge 1}\frac{1}{\|w\|} \iff \min_{y_i(w^\top x_i+b)\ge 1}\|w\| \iff \min_{y_i(w^\top x_i+b)\ge 1}\tfrac{1}{2}\|w\|^2.$$

Lagrangian and stationarity

Primal (hard-margin SVM):

$$\min_{w\in\mathbb{R}^d,\, b\in\mathbb{R}} \quad \tfrac{1}{2}\|w\|^2$$
$$\text{s.t.} \quad y_i\big(w^\top x_i+b\big)\ge 1, \qquad i=1,\dots,n.$$

Define constraint functions

$$g_i(w,b):=1-y_i\big(w^\top x_i+b\big)\le 0, \qquad i=1,\dots,n,$$

and Lagrange multipliers $\alpha_i\ge 0$.
Lagrangian:

$$\mathcal{L}(w,b,\alpha)=\tfrac{1}{2}w^\top w+\sum_{i=1}^n\alpha_i\big(1-y_i(w^\top x_i+b)\big).$$

Stationarity (set partial derivatives w.r.t. primal variables to zero):

w:

$$\frac{\partial\mathcal{L}}{\partial w}=w-\sum_{i=1}^n\alpha_i y_i x_i=0 \quad\Longrightarrow\quad \boxed{w=\sum_{i=1}^n\alpha_i y_i x_i\,.}$$

b:

$$\frac{\partial\mathcal{L}}{\partial b}=-\sum_{i=1}^n\alpha_i y_i=0 \quad\Longrightarrow\quad \boxed{\sum_{i=1}^n\alpha_i y_i=0\,.}$$

Constraints on dual variables:

$$\alpha_i\ge 0, \qquad i=1,\dots,n.$$

(End of subsection: Lagrangian defined and stationarity conditions derived.)

Dual form of the Lagrangian — derivation

Primal Lagrangian (repeated for clarity):

$$\mathcal{L}(w,b,\alpha)=\tfrac{1}{2}w^\top w+\sum_{i=1}^n\alpha_i\big(1-y_i(w^\top x_i+b)\big), \qquad \alpha_i\ge 0.$$

Collect terms in $\mathcal{L}$:

$$\mathcal{L}(w,b,\alpha)=\tfrac{1}{2}w^\top w+\sum_{i=1}^n\alpha_i-\sum_{i=1}^n\alpha_i y_i w^\top x_i-b\sum_{i=1}^n\alpha_i y_i$$
$$=\tfrac{1}{2}w^\top w-w^\top\Big(\sum_{i=1}^n\alpha_i y_i x_i\Big)+\sum_{i=1}^n\alpha_i-b\sum_{i=1}^n\alpha_i y_i.$$

Dual function definition:

$$d(\alpha)\ :=\ \inf_{w,b}\ \mathcal{L}(w,b,\alpha).$$

Minimization over b: the term $-b\sum_i\alpha_i y_i$ is finite iff $\sum_{i=1}^n\alpha_i y_i=0$. Enforce this equality (otherwise $d(\alpha)=-\infty$). Thus impose constraint

$$\sum_{i=1}^n\alpha_i y_i=0.$$

Minimization over w: for fixed α (and under the b-constraint),

$$\mathcal{L}(w,\alpha)=\tfrac{1}{2}w^\top w-w^\top s+\sum_{i=1}^n\alpha_i, \qquad s:=\sum_{i=1}^n\alpha_i y_i x_i.$$

This is a convex quadratic in w. Stationarity $\nabla_w\mathcal{L}=0$ gives $w=s$. Substitute $w=s$ to obtain the minimal value:

$$\inf_w\Big(\tfrac{1}{2}w^\top w-w^\top s\Big)=\tfrac{1}{2}s^\top s-s^\top s=-\tfrac{1}{2}s^\top s.$$

Compute $s^\top s$:

$$s^\top s = \Big(\sum_{i=1}^n \alpha_i y_i x_i\Big)^\top \Big(\sum_{j=1}^n \alpha_j y_j x_j\Big) = \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \, x_i^\top x_j.$$

Therefore, under the constraints $\alpha_i \geq 0$ and $\sum_i \alpha_i y_i = 0$,

$$\boxed{d(\alpha) \; = \; \sum_{i=1}^n \alpha_i \; - \; \tfrac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \, x_i^\top x_j.}$$

Dual problem (concise):

$$\begin{aligned} \max_{\alpha \in \mathbb{R}^n} \quad & \sum_{i=1}^n \alpha_i \; - \; \tfrac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \, x_i^\top x_j \\ \text{s.t.} \quad & \alpha_i \geq 0, \quad i = 1, \dots, n, \\ & \sum_{i=1}^n \alpha_i y_i = 0. \end{aligned}$$

Note: primal variable recovery uses stationarity $w = \sum_i \alpha_i y_i x_i$ and any support index s with $\alpha_s > 0$ gives $b = y_s - w^\top x_s$.

Soft Margin Support Vector Machines

Soft-Margin Primal Objective

$$\begin{aligned} \min_{\mathbf{w}, b, \boldsymbol{\xi}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad i = 1, \dots, N, \\ & \xi_i \geq 0, \quad i = 1, \dots, N. \end{aligned}$$

A. Interpretation of the Slack Variable ξ_i

From the constraints $y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$:

- If $\xi_i = 0$, then $y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1$. The point $\mathbf{x}_i$ is outside or on the margin (correctly classified, margin ≥ 1).
- If $0 < \xi_i < 1$, then $0 < y_i(\mathbf{w}^\top \mathbf{x}_i + b) < 1$. The point is correctly classified but lies *inside* the margin (margin violated).
- If $\xi_i \geq 1$, then $y_i(\mathbf{w}^\top \mathbf{x}_i + b) \leq 1 - \xi_i \leq 0$. The point is misclassified.

ξ_i is a nonnegative measure of margin violation.

B. Slack Equals Hinge Loss at Optimality

For a fixed $(\mathbf{w}, b)$, the smallest feasible ξ_i satisfying the two inequalities is:

$$\xi_i^\star(\mathbf{w}, b) = \max\{0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)\}.$$

Substituting this back into the primal objective eliminates $\boldsymbol{\xi}$ and yields the unconstrained hinge-loss formulation:

$$\min_{\mathbf{w}, b} \; \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^N \max\big(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)\big).$$

Thus, at optimality, the slack ξ_i is precisely the hinge loss for that point: $\xi_i = \ell_{\text{hinge}}(y_i f(\mathbf{x}_i))$.

C. Meaning of the Parameter C

The scalar $C > 0$ balances $\frac{1}{2}\|\mathbf{w}\|^2 + C \sum_i \xi_i$.

1. **Trade-off parameter:** C balances margin maximization (minimizing $\|\mathbf{w}\|$) against minimizing total margin violation (minimizing $\sum \xi_i$).
2. **Loss vs. regularization weight:** In the hinge-loss view, C is the multiplier for the empirical loss.
   - **Large** C: Prioritizes empirical loss (fit data, small $\sum \xi_i$), even at the cost of a narrower margin (larger $\|\mathbf{w}\|$). Risks overfitting.
   - **Small** C: Prioritizes regularization (wider margin, small $\|\mathbf{w}\|$), even if it means allowing larger violations. Risks underfitting.
3. **Dual interpretation:** In the dual problem, the multipliers α_i are constrained by $0 \leq \alpha_i \leq C$. C acts as an upper bound on the influence any single training point can have on the solution.
4. **Relationship to λ:** Some texts use $\min \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{N} \sum \ell_{\text{hinge}}$. C and λ are inversely related (e.g., $C = 1/(N\lambda)$). A big C corresponds to a small λ (weak regularization).

D. The Margin vs. Misclassification Trade-off

The optimizer chooses $(\mathbf{w}, b, \boldsymbol{\xi})$ to minimize the weighted sum $J = \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i$.

1. $C \to \infty$: Violations are punished infinitely. If data is separable, the solution converges to the hard-margin SVM.
2. $C \to 0^+$: Violations are almost free. The optimizer only minimizes $\|\mathbf{w}\|$, leading to $\mathbf{w} \approx \mathbf{0}$ (maximum margin), ignoring the data entirely.
3. Finite C: A balance is struck. Increasing C leads to smaller $\sum \xi_i$ (fewer violations) at the cost of a larger $\|\mathbf{w}\|$ (narrower margin).

KKT Analysis: The Lagrangian is:

$$\mathcal{L} = \tfrac{1}{2}\|\mathbf{w}\|^2 + C \sum_i \xi_i - \sum_i \alpha_i (y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 + \xi_i) - \sum_i \mu_i \xi_i$$

where $\alpha_i \geq 0$ and $\mu_i \geq 0$. Stationarity w.r.t. ξ_i yields:

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = C - \alpha_i - \mu_i = 0 \quad \Rightarrow \quad \alpha_i + \mu_i = C$$

Since $\mu_i \geq 0$, this immediately implies the box constraint $0 \leq \alpha_i \leq C$. The KKT complementary slackness conditions are:

$$\alpha_i\big(y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 + \xi_i\big) = 0 \quad \text{and} \quad \mu_i \xi_i = 0$$

These equations define the role of every data point:

- $\alpha_i = 0$: Then $\mu_i = C > 0$, which forces $\xi_i = 0$. The α_i condition implies $y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1$. Point is correctly classified and outside the margin.
- $0 < \alpha_i < C$: Then $\mu_i = C - \alpha_i > 0$, which forces $\xi_i = 0$. The α_i condition forces $y_i(\mathbf{w}^\top \mathbf{x}_i + b) = 1$. Point is a support vector exactly on the margin.
- $\alpha_i = C$: Then $\mu_i = 0$, so ξ_i is free to be ≥ 0. The α_i condition forces $y_i(\mathbf{w}^\top \mathbf{x}_i + b) = 1 - \xi_i$. Point is a margin-violator, either inside the margin ($\xi_i < 1$) or misclassified ($\xi_i \geq 1$).

E. Concrete Numeric/Logic Summary

- $\xi_i = 0 \Leftrightarrow$ point is correct and on or outside margin.
- $0 < \xi_i < 1 \Leftrightarrow$ point is correct but inside margin.
- $\xi_i \geq 1 \Leftrightarrow$ point is misclassified.
- Larger $C \to$ fewer/smaller violations $\to$ fits training data more (overfit risk).
- Smaller $C \to$ allows violations $\to$ keeps $\|\mathbf{w}\|$ small (underfit risk).

Final Compact Takeaway

- Slack ξ_i measures how much the i-th point violates the desired margin of 1. It equals the hinge loss value for that point at optimality.
- Parameter C controls the penalty for violations; it is the trade-off knob between margin size (model complexity) and misclassification (training error).

Derivation of the Soft-Margin Lagrangian

We start with the soft-margin primal problem:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ \text{s.t.} \quad & y_i\big(\mathbf{w}^\top\mathbf{x}_i + b\big) \geq 1 - \xi_i, \quad i = 1,\dots,N, \\ & \xi_i \geq 0, \quad i = 1,\dots,N. \end{aligned}$$

We write the constraints in the canonical form $g_i(\cdot) \leq 0$:

$$\begin{aligned} g_i(\mathbf{w}, b, \xi) &:= 1 - \xi_i - y_i(\mathbf{w}^\top\mathbf{x}_i + b) \leq 0 \\ h_i(\mathbf{w}, b, \xi) &:= -\xi_i \leq 0 \end{aligned}$$

Introduce Lagrange multipliers $\alpha_i \geq 0$ for g_i and $\mu_i \geq 0$ for h_i. The Lagrangian $\mathcal{L}$ is:

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i + \sum_{i=1}^{N}\alpha_i\big(1 - \xi_i - y_i(\mathbf{w}^\top\mathbf{x}_i + b)\big) - \sum_{i=1}^{N}\mu_i\xi_i$$

Expand and collect terms:

$$\begin{aligned} \mathcal{L} &= \frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\sum_{i=1}^{N}\xi_i + \sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{N}\alpha_i\xi_i - \sum_{i=1}^{N}\alpha_i y_i\mathbf{w}^\top\mathbf{x}_i - b\sum_{i=1}^{N}\alpha_i y_i - \sum_{i=1}^{N}\mu_i\xi_i \\ &= \frac{1}{2}\mathbf{w}^\top\mathbf{w} - \mathbf{w}^\top\Big(\sum_{i=1}^{N}\alpha_i y_i\mathbf{x}_i\Big) - b\Big(\sum_{i=1}^{N}\alpha_i y_i\Big) + \sum_{i=1}^{N}\alpha_i + \sum_{i=1}^{N}\xi_i(C - \alpha_i - \mu_i) \end{aligned}$$

Weight vector

The optimal weight vector $\mathbf{w}$ can be expressed entirely in terms of the data points $\mathbf{x}_i$ and the dual variables α_i.

Derivation from the Lagrangian

We start with the Lagrangian (for both soft and hard margin SVMs). The terms that involve $\mathbf{w}$ are:

$$\mathcal{L}(\mathbf{w}, \dots) = \frac{1}{2}\mathbf{w}^\top\mathbf{w} - \sum_{i=1}^{N}\alpha_i y_i\mathbf{w}^\top\mathbf{x}_i + (\text{terms not involving } \mathbf{w}).$$

To find the optimal $\mathbf{w}$ for the dual problem, we minimize $\mathcal{L}$ with respect to the primal variables. We take the gradient of $\mathcal{L}$ with respect to $\mathbf{w}$:

$$\nabla_{\mathbf{w}}\mathcal{L} = \frac{\partial}{\partial\mathbf{w}}\left(\frac{1}{2}\mathbf{w}^\top\mathbf{w} - \sum_{i=1}^{N}\alpha_i y_i\mathbf{w}^\top\mathbf{x}_i\right)$$

Using basic vector calculus:

- $\nabla_{\mathbf{w}}(\frac{1}{2}\mathbf{w}^\top\mathbf{w}) = \mathbf{w}$
- $\nabla_{\mathbf{w}}(\sum_i \alpha_i y_i\mathbf{w}^\top\mathbf{x}_i) = \nabla_{\mathbf{w}}\left(\mathbf{w}^\top\sum_i \alpha_i y_i\mathbf{x}_i\right) = \sum_i \alpha_i y_i\mathbf{x}_i$

This gives:

$$\nabla_{\mathbf{w}}\mathcal{L} = \mathbf{w} - \sum_{i=1}^{N}\alpha_i y_i\mathbf{x}_i$$

The stationarity condition (KKT) requires this gradient to be zero:

$$\nabla_{\mathbf{w}}\mathcal{L} = 0 \quad \Rightarrow \quad \mathbf{w} - \sum_{i=1}^{N}\alpha_i y_i\mathbf{x}_i = 0$$

Rearranging yields the expression for $\mathbf{w}$:

$$\boxed{\mathbf{w} = \sum_{i=1}^{N}\alpha_i y_i\mathbf{x}_i}$$

This shows that the optimal weight vector $\mathbf{w}$ is a linear combination of the feature vectors of the support vectors (those $\mathbf{x}_i$ for which $\alpha_i > 0$).

Selection of Support Vectors

The KKT conditions provide a precise mathematical procedure for identifying support vectors from the optimal dual variables $\boldsymbol{\alpha}^\star$.

1. Hard-Margin SVM

Dual (hard-margin):

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i^\top\mathbf{x}_j) \\ \text{s.t.} \quad & \alpha_i \geq 0, \quad \sum_{i=1}^{N}\alpha_i y_i = 0. \end{aligned}$$

KKT Complementary Slackness: The primal constraint is $y_i(\mathbf{w}^\top\mathbf{x}_i + b) \geq 1$. The KKT condition is:

$$\alpha_i\big(y_i(\mathbf{w}^\top\mathbf{x}_i + b) - 1\big) = 0.$$

Interpretation:

- If $\alpha_i = 0$: The constraint is inactive. This implies $y_i(\mathbf{w}^\top\mathbf{x}_i + b) > 1$. The point is strictly outside the margin and is **not a support vector**.
- If $\alpha_i > 0$: The constraint must be active. This forces $y_i(\mathbf{w}^\top\mathbf{x}_i + b) = 1$. The point lies exactly on the margin and is a **support vector**.

Hard-Margin Support Vector Selection Rule:

$$\boxed{\text{Support vectors are exactly the indices } i \text{ such that } \alpha_i > 0.}$$

2. Soft-Margin SVM

Dual (soft-margin):

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{s.t.} \quad & 0 \le \alpha_i \le C, \quad \sum_{i=1}^{N} \alpha_i y_i = 0. \end{aligned}$$

Key KKT Relations:

$$\alpha_i(y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 + \xi_i) = 0 \qquad \text{(Primal slackness)}$$

$$\mu_i \xi_i = 0 \qquad \text{(Dual slackness)}$$

$$\alpha_i + \mu_i = C \qquad \text{(Stationarity w.r.t } \xi_i\text{)}$$

Soft-margin point categories from KKT: Every training point falls into one of three categories:

- **Case 1:** $\alpha_i = 0$ **(Non-Support Vector)**
  From $\alpha_i + \mu_i = C$, we get $\mu_i = C > 0$. From $\mu_i \xi_i = 0$, this forces $\xi_i = 0$. The primal slackness condition $\alpha_i(\dots) = 0$ is satisfied. The point satisfies $y_i(\mathbf{w}^\top \mathbf{x}_i + b) > 1$. **Result:** Point is strictly outside the margin. Not a support vector.
- **Case 2:** $0 < \alpha_i < C$ **(Margin Support Vector)**
  From $\alpha_i + \mu_i = C$, we get $\mu_i = C - \alpha_i > 0$. From $\mu_i \xi_i = 0$, this forces $\xi_i = 0$. The primal slackness condition $\alpha_i(\dots) = 0$ forces $y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 + 0 = 0$. **Result:** $y_i(\mathbf{w}^\top \mathbf{x}_i + b) = 1$. Point lies exactly on the margin. **This is a support vector**.
- **Case 3:** $\alpha_i = C$ **(Violating Support Vector)**
  From $\alpha_i + \mu_i = C$, we get $\mu_i = 0$. Since $\mu_i = 0$, the condition $\mu_i \xi_i = 0$ is satisfied, and ξ_i is free to be ≥ 0. The primal slackness condition $\alpha_i(\dots) = 0$ forces $y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 + \xi_i = 0$. **Result:** $y_i(\mathbf{w}^\top \mathbf{x}_i + b) = 1 - \xi_i \le 1$. The point is either inside the margin (if $0 < \xi_i < 1$) or misclassified (if $\xi_i \ge 1$). **This is a support vector**.

Soft-Margin Support Vector Selection Rule:

$$\boxed{\text{Support vectors are all indices } i \text{ such that } \alpha_i > 0.}$$

3. Final Summary Table

4. Practical Mathematical Procedure

Given the optimal dual solution $\boldsymbol{\alpha}^\star$:

1. Compute the set of all support vectors:

$$S = \{i : \alpha_i^\star > 0\}$$

2. Partition S into margin and violating SVs:
   - Margin SVs: $S_{\text{margin}} = \{i \in S : 0 < \alpha_i^\star < C\}$
   - Violating SVs: $S_{\text{error}} = \{i \in S : \alpha_i^\star = C\}$
3. Non-SVs are the remaining points:

$$S_{\text{non}} = \{i : \alpha_i^\star = 0\}$$

Chapter 7

Kernels

Kernel Functions

A kernel is a function that allows us to compute inner products in a high-dimensional feature space without explicitly mapping to that space. This is known as the ”kernel trick”.

Definition of a Kernel Function

A kernel function is a function $K : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ such that there exists a mapping $\phi : \mathbb{R}^D \to \mathcal{H}$ into some (possibly high-dimensional or infinite-dimensional) Hilbert space $\mathcal{H}$ satisfying:

$$\boxed{K(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle_{\mathcal{H}}}$$

This allows us to compute the inner product in $\mathcal{H}$ without ever needing to compute $\phi(\mathbf{x})$ explicitly.

Geometric Interpretation of a Kernel

1. Kernels compute dot products in a transformed space. Even if your input $\mathbf{x}$ lives in $\mathbb{R}^D$, the kernel behaves as if you mapped the data into an enormous space $\mathcal{H}$ via $\mathbf{x} \mapsto \phi(\mathbf{x})$. Instead of computing $\phi(\mathbf{x})$, the kernel gives the geometric quantity needed:

$$K(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle.$$

Geometrically, $K(\mathbf{x}, \mathbf{z})$ measures the similarity (angle, orientation) of the points in the feature space $\mathcal{H}$. The margin and separating hyperplane in SVMs are computed in $\mathcal{H}$, not the original input space.

2. Kernel = Similarity Measure in Feature Space. The kernel value corresponds to:

$$K(\mathbf{x}, \mathbf{z}) = \|\phi(\mathbf{x})\|_{\mathcal{H}} \cdot \|\phi(\mathbf{z})\|_{\mathcal{H}} \cdot \cos(\theta),$$

where θ is the angle between the transformed vectors.

- Large $K(\mathbf{x}, \mathbf{z}) \to$ points are geometrically similar in $\mathcal{H}$.
- Small/Zero $K(\mathbf{x}, \mathbf{z}) \to$ points are orthogonal or dissimilar in $\mathcal{H}$.

3. Kernel defines the SVM decision boundary. The SVM classifier in the dual (kernelized) form is:

$$f(\mathbf{x}) = \sum_{i \in \text{SV}} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b.$$

Geometrically, the decision function is a weighted sum of similarities between the test point $\mathbf{x}$ and the support vectors $\mathbf{x}_i$ in the feature space. The separating hyperplane is linear in $\mathcal{H}$ but nonlinear in $\mathbb{R}^D$.

4. Kernel implicitly defines the geometry. Different kernels correspond to different geometries for the decision boundary in the original space:

- **Linear:** $K(\mathbf{x}, \mathbf{z}) = \mathbf{x}^\top \mathbf{z}$. Original space inner product.
- **Polynomial:** $K(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\top \mathbf{z} + c)^p$. Space of monomials up to degree p.
- **RBF / Gaussian:** $K(\mathbf{x}, \mathbf{z}) = \exp\big(-\gamma \|\mathbf{x} - \mathbf{z}\|^2\big)$. Infinite-dimensional feature space; distance-based similarity.

Final One-Line Summary

A kernel is a function that computes inner products in a high-dimensional feature space, allowing SVMs to construct linear geometric separators in that feature space, which are nonlinear in the original input space.

Mercer's Theorem (Kernel Validity Check)

Mercer's theorem provides a practical way to check if a function $K(\mathbf{x}, \mathbf{z})$ is a valid kernel.

Practical Definition: A symmetric function, $K(\mathbf{x}, \mathbf{z}) = K(\mathbf{z}, \mathbf{x})$, is a valid kernel if and only if its Gram matrix is positive semidefinite (PSD) for every finite dataset.

Step-by-step check:

1. Pick any finite set of points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$.
2. Form the $N \times N$ Gram matrix $\mathbf{K}$ where $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$.
3. Check if $\mathbf{K}$ is positive semidefinite. This means for any vector $\mathbf{c} \in \mathbb{R}^N$:

   $$\mathbf{c}^\top \mathbf{K} \mathbf{c} \geq 0$$

   (Equivalently, all eigenvalues of $\mathbf{K}$ must be ≥ 0).

Concise Mercer Test:

$$\boxed{K(\mathbf{x}, \mathbf{z}) \text{ is a kernel} \iff \mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j) \text{ is PSD for every finite set } \{\mathbf{x}_i\}.}$$

Why this works: Mercer's theorem (in one form) states that K is a valid kernel if and only if

$$\iint f(\mathbf{x}) K(\mathbf{x}, \mathbf{z}) f(\mathbf{z}) \, d\mathbf{x} \, d\mathbf{z} \geq 0 \quad \forall f$$

Checking that the Gram matrix $\mathbf{K}$ is PSD for every finite set of points is the discrete equivalent of this continuous condition.

Verifying Kernels via Feature Maps (PSD Proofs)

A direct way to prove a kernel $K(\mathbf{x},\mathbf{z})$ is positive semidefinite (PSD) is to find an explicit feature map ϕ such that $K(\mathbf{x},\mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z})\rangle$. If such a ϕ exists, Mercer's theorem is automatically satisfied.

Example 1: Linear Kernel $K(\mathbf{x},\mathbf{z}) = \mathbf{x}^\top\mathbf{z}$

We show K is PSD by showing its Gram matrix $\mathbf{K}$ is PSD. By definition, $\mathbf{K}$ is PSD iff for every finite set $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ and every $\mathbf{c} \in \mathbb{R}^N$:

$$\mathbf{c}^\top \mathbf{K}\mathbf{c} = \sum_{i=1}^{N}\sum_{j=1}^{N} c_i c_j K(\mathbf{x}_i, \mathbf{x}_j) \geq 0.$$

For the linear kernel, the feature map is simply $\phi(\mathbf{x}) = \mathbf{x}$.

$$\begin{aligned}
\mathbf{c}^\top \mathbf{K}\mathbf{c} &= \sum_{i=1}^{N}\sum_{j=1}^{N} c_i c_j (\mathbf{x}_i^\top \mathbf{x}_j) \\
&= \sum_{i=1}^{N} c_i \mathbf{x}_i^\top \sum_{j=1}^{N} c_j \mathbf{x}_j \\
&= \left(\sum_{i=1}^{N} c_i \mathbf{x}_i\right)^\top \left(\sum_{j=1}^{N} c_j \mathbf{x}_j\right) \\
&= \left\|\sum_{i=1}^{N} c_i \mathbf{x}_i\right\|^2 \geq 0.
\end{aligned}$$

The result is a squared norm, which is non-negative. Thus the linear kernel is PSD.

Example 2: Polynomial Kernel $K(\mathbf{x},\mathbf{z}) = (\mathbf{x}^\top\mathbf{z} + 1)^2$

Let $\mathbf{x}, \mathbf{z} \in \mathbb{R}^D$. First, expand the kernel:

$$K(\mathbf{x},\mathbf{z}) = (\mathbf{x}^\top\mathbf{z})^2 + 2(\mathbf{x}^\top\mathbf{z}) + 1.$$

We now construct an explicit feature map ϕ. We need to list all degree-2 cross terms, degree-2 squared terms, degree-1 terms, and the constant term. For $\mathbf{x} \in \mathbb{R}^D$, define $\phi(\mathbf{x})$ as the vector:

$$\phi(\mathbf{x}) = (\underbrace{x_1^2, \dots, x_D^2}_{\text{D terms}}, \underbrace{\sqrt{2}x_i x_j}_{\text{for } i<j}, \underbrace{\sqrt{2}x_1, \dots, \sqrt{2}x_D}_{\text{D terms}}, 1)$$

Now we compute the inner product $\langle \phi(\mathbf{x}), \phi(\mathbf{z})\rangle$:

$$\begin{aligned}
\langle \phi(\mathbf{x}), \phi(\mathbf{z})\rangle &= \sum_{i=1}^{D}(x_i^2 z_i^2) + \sum_{i<j}(\sqrt{2}x_i x_j)(\sqrt{2}z_i z_j) + \sum_{i=1}^{D}(\sqrt{2}x_i)(\sqrt{2}z_i) + 1\cdot 1 \\
&= \sum_{i=1}^{D} x_i^2 z_i^2 + 2\sum_{i<j} x_i z_i x_j z_j + 2\sum_{i=1}^{D} x_i z_i + 1 \\
&= \left(\sum_{i=1}^{D} x_i z_i\right)^2 + 2\left(\sum_{i=1}^{D} x_i z_i\right) + 1 \\
&= (\mathbf{x}^\top\mathbf{z})^2 + 2(\mathbf{x}^\top\mathbf{z}) + 1 \\
&= (\mathbf{x}^\top\mathbf{z} + 1)^2 = K(\mathbf{x},\mathbf{z}).
\end{aligned}$$

Since we have found an explicit feature map ϕ such that $K(\mathbf{x},\mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z})\rangle$, the kernel is PSD by the proof in the Mercer's Theorem section (where $\mathbf{c}^\top\mathbf{K}\mathbf{c} = \|\sum_i c_i \phi(\mathbf{x}_i)\|^2 \geq 0$).

The RBF (Gaussian) Kernel

The RBF/Gaussian kernel $K(\mathbf{x},\mathbf{z}) = \exp\bigl(-\gamma\|\mathbf{x}-\mathbf{z}\|^2\bigr)$ (where $\gamma = 1/(2\sigma^2)$) corresponds to an inner product in an **infinite-dimensional** feature space. We can show this by constructing the explicit feature map.

$$K(\mathbf{x},\mathbf{z}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{z}\|^2}{2\sigma^2}\right)$$

1. Algebraic Rearrangement

We expand the squared norm $\|\mathbf{x}-\mathbf{z}\|^2 = \|\mathbf{x}\|^2 + \|\mathbf{z}\|^2 - 2\mathbf{x}^\top\mathbf{z}$.

$$K(\mathbf{x},\mathbf{z}) = \exp\left(-\frac{\|\mathbf{x}\|^2}{2\sigma^2}\right)\exp\left(-\frac{\|\mathbf{z}\|^2}{2\sigma^2}\right)\exp\left(\frac{\mathbf{x}^\top\mathbf{z}}{\sigma^2}\right)$$

To find the feature map, we only need to expand the cross-term $\exp\bigl(\mathbf{x}^\top\mathbf{z}/\sigma^2\bigr)$, as the other terms can be factored into $\langle \phi(\mathbf{x}), \phi(\mathbf{z})\rangle$.

2. Power-Series Expansion

Using the Taylor series $\exp(t) = \sum_{k=0}^{\infty}\frac{t^k}{k!}$ with $t = \mathbf{x}^\top\mathbf{z}/\sigma^2$:

$$\exp\left(\frac{\mathbf{x}^\top\mathbf{z}}{\sigma^2}\right) = \sum_{k=0}^{\infty}\frac{1}{k!}\frac{(\mathbf{x}^\top\mathbf{z})^k}{\sigma^{2k}}$$

Each monomial $(\mathbf{x}^\top\mathbf{z})^k$ can be expanded using the multinomial theorem. Let $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_D)$ be a multi-index with $|\boldsymbol{\alpha}| = \sum_i \alpha_i = k$. Let

$$m_{\boldsymbol{\alpha}}(\mathbf{x}) = \sqrt{\frac{k!}{\boldsymbol{\alpha}!}}x_1^{\alpha_1}\cdots x_D^{\alpha_D}$$

where $\boldsymbol{\alpha}! = \alpha_1!\cdots\alpha_D!$. Then the multinomial expansion gives $(\mathbf{x}^\top\mathbf{z})^k = \sum_{|\boldsymbol{\alpha}|=k} m_{\boldsymbol{\alpha}}(\mathbf{x})m_{\boldsymbol{\alpha}}(\mathbf{z})$. Substituting this in:

$$\exp\left(\frac{\mathbf{x}^\top\mathbf{z}}{\sigma^2}\right) = \sum_{k=0}^{\infty}\frac{1}{k!\sigma^{2k}}\sum_{|\boldsymbol{\alpha}|=k} m_{\boldsymbol{\alpha}}(\mathbf{x})m_{\boldsymbol{\alpha}}(\mathbf{z})$$

3. Assembling the Infinite-Dimensional Feature Map

Combine the Gaussian prefactors and the series:

$$\begin{aligned} K(\mathbf{x},\mathbf{z}) &= \exp\left(-\frac{\|\mathbf{x}\|^2}{2\sigma^2}\right)\exp\left(-\frac{\|\mathbf{z}\|^2}{2\sigma^2}\right)\sum_{k=0}^{\infty}\frac{1}{k!\sigma^{2k}}\sum_{|\boldsymbol{\alpha}|=k} m_{\boldsymbol{\alpha}}(\mathbf{x})m_{\boldsymbol{\alpha}}(\mathbf{z}) \\ &= \sum_{k=0}^{\infty}\sum_{|\boldsymbol{\alpha}|=k}\left(\exp\left(-\frac{\|\mathbf{x}\|^2}{2\sigma^2}\right)\frac{m_{\boldsymbol{\alpha}}(\mathbf{x})}{\sigma^k\sqrt{k!}}\right)\left(\exp\left(-\frac{\|\mathbf{z}\|^2}{2\sigma^2}\right)\frac{m_{\boldsymbol{\alpha}}(\mathbf{z})}{\sigma^k\sqrt{k!}}\right) \end{aligned}$$

We can define the feature map $\phi(\mathbf{x})$ as the infinite vector whose components are indexed by k and $\boldsymbol{\alpha}$ (with $|\boldsymbol{\alpha}| = k$):

$$\phi_{k,\boldsymbol{\alpha}}(\mathbf{x}) := \exp\left(-\frac{\|\mathbf{x}\|^2}{2\sigma^2}\right)\frac{m_{\boldsymbol{\alpha}}(\mathbf{x})}{\sigma^k\sqrt{k!}}$$

With this definition, we have shown:

$$K(\mathbf{x},\mathbf{z}) = \langle\phi(\mathbf{x}),\phi(\mathbf{z})\rangle_{\ell^2}$$

This is an inner product in an infinite-dimensional Hilbert space ℓ^2 (the space of square-summable sequences).

4. Why this proves Infinite Dimensionality

For each degree k, there are $\binom{D+k-1}{k}$ distinct multi-indices $\boldsymbol{\alpha}$. As k ranges over all non-negative integers $(0,1,2,\dots)$, the total number of components is infinite. Since the Taylor series for $\exp(t)$ contains non-zero terms for every k, infinitely many feature components are non-zero. No finite-length vector $\phi(\mathbf{x}) \in \mathbb{R}^M$ can reproduce this kernel for all $\mathbf{x},\mathbf{z}$.

5. Intuition

The RBF kernel is a weighted sum of inner products of all homogeneous monomials of all degrees (with Gaussian damping), so it behaves as if data were embedded into a space containing all polynomial features, which is infinite-dimensional.

Cosine Similarity Kernel

We analyze the kernel $K(\mathbf{x},\mathbf{z}) = \frac{\mathbf{x}^\top\mathbf{z}}{\|\mathbf{x}\|\|\mathbf{z}\|}$, for $\mathbf{x},\mathbf{z} \in \mathbb{R}^D \setminus \{\mathbf{0}\}$.

Feature Map

We can define an explicit feature map ϕ that normalizes the input vectors:

$$\phi : \mathbb{R}^D \setminus \{\mathbf{0}\} \to \mathbb{R}^D, \qquad \phi(\mathbf{x}) = \frac{\mathbf{x}}{\|\mathbf{x}\|}$$

Then for all non-zero $\mathbf{x},\mathbf{z}$, the kernel is the inner product of these normalized vectors:

$$K(\mathbf{x},\mathbf{z}) = \frac{\mathbf{x}^\top\mathbf{z}}{\|\mathbf{x}\|\|\mathbf{z}\|} = \left\langle\frac{\mathbf{x}}{\|\mathbf{x}\|},\frac{\mathbf{z}}{\|\mathbf{z}\|}\right\rangle = \langle\phi(\mathbf{x}),\phi(\mathbf{z})\rangle$$

(This can be extended to handle zero vectors by defining $\phi(\mathbf{0}) = \mathbf{0}$ and $K(\mathbf{0},\mathbf{z}) = 0$.)

Proof of PSD

Since we have found an explicit feature map ϕ, we can use the standard proof. Take any finite set $\{\mathbf{x}_1,\dots,\mathbf{x}_N\} \subset \mathbb{R}^D \setminus \{\mathbf{0}\}$ and any $\mathbf{c} \in \mathbb{R}^N$. Form the Gram matrix $\mathbf{K}$ with entries $K_{ij} = K(\mathbf{x}_i,\mathbf{x}_j) = \langle\phi(\mathbf{x}_i),\phi(\mathbf{x}_j)\rangle$. Then:

$$\begin{aligned} \mathbf{c}^\top\mathbf{K}\mathbf{c} &= \sum_{i=1}^{N}\sum_{j=1}^{N} c_i c_j \langle\phi(\mathbf{x}_i),\phi(\mathbf{x}_j)\rangle \\ &= \left\langle\sum_{i=1}^{N} c_i\phi(\mathbf{x}_i),\sum_{j=1}^{N} c_j\phi(\mathbf{x}_j)\right\rangle \\ &= \left\|\sum_{i=1}^{N} c_i\phi(\mathbf{x}_i)\right\|^2 \geq 0 \end{aligned}$$

The Gram matrix is positive semidefinite for every finite set. The function is also symmetric since $K(\mathbf{x},\mathbf{z}) = K(\mathbf{z},\mathbf{x})$.

Conclusion

The cosine similarity function is a valid positive semidefinite kernel.

Constructing Kernels

We can prove a function is a valid PSD kernel by constructing its feature map.

1. Feature Map for the Sum of Kernels

If $K_1(\mathbf{x},\mathbf{z}) = \langle\phi_1(\mathbf{x}),\phi_1(\mathbf{z})\rangle_{\mathcal{H}_1}$ and $K_2(\mathbf{x},\mathbf{z}) = \langle\phi_2(\mathbf{x}),\phi_2(\mathbf{z})\rangle_{\mathcal{H}_2}$ are valid kernels, then their sum $K = K_1 + K_2$ is also a valid kernel.

We define a new feature map ϕ by concatenating the individual feature maps:

$$\phi(\mathbf{x}) = \big(\phi_1(\mathbf{x}),\phi_2(\mathbf{x})\big) \in \mathcal{H}_1 \oplus \mathcal{H}_2$$

This new vector lives in the direct sum of the two Hilbert spaces. The inner product in this combined space is the sum of the inner products in the component spaces.

$$\begin{aligned} (K_1 + K_2)(\mathbf{x},\mathbf{z}) &= K_1(\mathbf{x},\mathbf{z}) + K_2(\mathbf{x},\mathbf{z}) \\ &= \langle\phi_1(\mathbf{x}),\phi_1(\mathbf{z})\rangle_{\mathcal{H}_1} + \langle\phi_2(\mathbf{x}),\phi_2(\mathbf{z})\rangle_{\mathcal{H}_2} \\ &= \big\langle(\phi_1(\mathbf{x}),\phi_2(\mathbf{x})),(\phi_1(\mathbf{z}),\phi_2(\mathbf{z}))\big\rangle_{\mathcal{H}_1\oplus\mathcal{H}_2} \\ &= \langle\phi(\mathbf{x}),\phi(\mathbf{z})\rangle_{\mathcal{H}_1\oplus\mathcal{H}_2} \end{aligned}$$

Since $K_1 + K_2$ is an inner product, it is a valid PSD kernel.

2. Feature Map for $K(\mathbf{x},\mathbf{z}) = \exp(\mathbf{x}^\top\mathbf{z})$

We use the exponential power series and the multinomial expansion. First, the Taylor series for $\exp(t)$:

$$\exp(\mathbf{x}^\top\mathbf{z}) = \sum_{k=0}^{\infty}\frac{(\mathbf{x}^\top\mathbf{z})^k}{k!}$$

Next, the multinomial expansion for $(\mathbf{x}^\top\mathbf{z})^k$:

$$(\mathbf{x}^\top\mathbf{z})^k = \left(\sum_{i=1}^{D} x_i z_i\right)^k = \sum_{|\boldsymbol{\alpha}|=k}\frac{k!}{\boldsymbol{\alpha}!}\mathbf{x}^{\boldsymbol{\alpha}}\mathbf{z}^{\boldsymbol{\alpha}}$$

where $\boldsymbol{\alpha} = (\alpha_1,\dots,\alpha_D)$ is a multi-index, $|\boldsymbol{\alpha}| = \sum_i \alpha_i = k$, $\boldsymbol{\alpha}! = \prod_i \alpha_i!$, and $\mathbf{x}^{\boldsymbol{\alpha}} = \prod_i x_i^{\alpha_i}$.

Substitute the expansion into the series:

$$\begin{aligned}\exp(\mathbf{x}^\top\mathbf{z}) &= \sum_{k=0}^{\infty}\frac{1}{k!}\sum_{|\boldsymbol{\alpha}|=k}\frac{k!}{\boldsymbol{\alpha}!}\mathbf{x}^{\boldsymbol{\alpha}}\mathbf{z}^{\boldsymbol{\alpha}} \\ &= \sum_{k=0}^{\infty}\sum_{|\boldsymbol{\alpha}|=k}\frac{\mathbf{x}^{\boldsymbol{\alpha}}}{\sqrt{\boldsymbol{\alpha}!}}\cdot\frac{\mathbf{z}^{\boldsymbol{\alpha}}}{\sqrt{\boldsymbol{\alpha}!}}\end{aligned}$$

This is an inner product. We can define an explicit (infinite-dimensional) feature map ϕ whose components are indexed by all possible multi-indices $\boldsymbol{\alpha}$:

$$\boxed{\phi_{\boldsymbol{\alpha}}(\mathbf{x}) = \frac{\mathbf{x}^{\boldsymbol{\alpha}}}{\sqrt{\boldsymbol{\alpha}!}}}$$

With this ϕ, we have:

$$\exp(\mathbf{x}^\top\mathbf{z}) = \sum_{\boldsymbol{\alpha}}\phi_{\boldsymbol{\alpha}}(\mathbf{x})\phi_{\boldsymbol{\alpha}}(\mathbf{z}) = \langle\phi(\mathbf{x}),\phi(\mathbf{z})\rangle_{\ell^2}$$

Since $K(\mathbf{x},\mathbf{z})$ is an inner product in an infinite-dimensional Hilbert space, it is a valid PSD kernel.

Kernelisation of Linear Models

3. The Representer Theorem

A standard result (the Representer Theorem) shows that any minimizer $\mathbf{w}^\star$ of the above problem admits an expansion as a linear combination of the feature-mapped training data:

Representer Theorem

$$\mathbf{w}^\star = \sum_{i=1}^{N}\alpha_i\phi(\mathbf{x}_i)$$

for some coefficients $\boldsymbol{\alpha}\in\mathbb{R}^N$. (Proof idea: Decompose any $\mathbf{w}$ into a component in $\mathrm{span}(\{\phi(\mathbf{x}_i)\})$ and an orthogonal component. The orthogonal component only increases the regularizer $\|\mathbf{w}\|^2$ without improving the empirical loss, so it must be zero at the minimum.)

4. Kernelized Decision Function

We derive the kernelized decision function by plugging the representation of $\mathbf{w}^\star$ from the Representer Theorem back into the linear model:

$$\begin{aligned}f(\mathbf{x}) &= (\mathbf{w}^\star)^\top\phi(\mathbf{x}) + b \\ &= \left(\sum_{i=1}^{N}\alpha_i\phi(\mathbf{x}_i)\right)^\top\phi(\mathbf{x}) + b \\ &= \sum_{i=1}^{N}\alpha_i\langle\phi(\mathbf{x}_i),\phi(\mathbf{x})\rangle + b\end{aligned}$$

Now, we replace the inner product with the kernel function $K(\mathbf{x}_i,\mathbf{x})$:

Kernelized Decision Function

$$f(\mathbf{x}) = \sum_{i=1}^{N}\alpha_i K(\mathbf{x}_i,\mathbf{x}) + b$$

This decision function only requires evaluations of the kernel K, never the explicit feature map ϕ.

5. Examples of Kernelized Models

(A) SVM Stationarity (Hard/Soft Margin) As shown previously, the KKT stationarity condition for SVMs yields $\mathbf{w} = \sum_i \alpha_i y_i\phi(\mathbf{x}_i)$. The prediction function becomes:

$$f(\mathbf{x}) = \mathbf{w}^\top\phi(\mathbf{x}) + b = \sum_i \alpha_i y_i K(\mathbf{x}_i,\mathbf{x}) + b$$

Here, the coefficients α_i are the dual variables (many of which are zero).

(B) Kernel Ridge Regression The primal problem in $\mathcal{H}$ is:

$$\min_{\mathbf{w}}\frac{1}{N}\|\mathbf{y}-\boldsymbol{\Phi}\mathbf{w}\|^2 + \lambda\|\mathbf{w}\|^2$$

where $\boldsymbol{\Phi}$ is the $N\times\dim(\mathcal{H})$ matrix of feature vectors $\phi(\mathbf{x}_i)^\top$. Using the Representer Theorem ($\mathbf{w} = \boldsymbol{\Phi}^\top\boldsymbol{\alpha}$) and solving for $\boldsymbol{\alpha}$ gives:

$$\boldsymbol{\alpha} = (\mathbf{K} + N\lambda\mathbf{I})^{-1}\mathbf{y}$$

where $\mathbf{K}$ is the $N\times N$ Gram matrix ($K_{ij} = K(\mathbf{x}_i,\mathbf{x}_j)$). The prediction is:

$$f(\mathbf{x}) = \sum_i \alpha_i K(\mathbf{x}_i,\mathbf{x})$$

6. Practical Consequences

- We can learn nonlinear decision boundaries by choosing a nonlinear kernel (Polynomial, RBF, etc.).
- All training computations reduce to operations on the $N\times N$ Gram matrix $\mathbf{K}$.
- Prediction complexity is $O(N)$ per test point, as we must compute the kernel between $\mathbf{x}$ and all N training points (or just the support vectors if $\boldsymbol{\alpha}$ is sparse, as in SVM).

Kernelisation via the Representer Theorem

The Representer Theorem is a powerful result that justifies kernelising many linear models. It states that for any regularized risk minimization problem of the form:

$$\min_{\mathbf{w}\in\mathcal{H}} L(\text{data},\mathbf{w}) + \Omega(\|\mathbf{w}\|^2)$$

where Ω is a strictly increasing function (e.g., L2 regularization $\frac{\lambda}{2}\|\mathbf{w}\|^2$), the optimal solution $\mathbf{w}^\star$ must lie in the span of the feature-mapped training data.

Representer Theorem

The optimal $\mathbf{w}^\star$ can be written as:

$$\mathbf{w}^\star = \sum_{j=1}^{N}\alpha_j\phi(\mathbf{x}_j) = \boldsymbol{\Phi}^\top\boldsymbol{\alpha}$$

where $\boldsymbol{\alpha}$ is a vector of N coefficients. This theorem allows us to convert the optimization problem from finding an (often infinite-dimensional) $\mathbf{w}$ to finding a finite, N-dimensional vector $\boldsymbol{\alpha}$.

1. Kernel Ridge Regression (KRR)

Primal Problem: KRR minimizes the L2-regularized squared loss in the feature space $\mathcal{H}$:

$$J(\mathbf{w}) = \sum_{i=1}^{N} (y_i - \mathbf{w}^\top \phi(\mathbf{x}_i))^2 + \lambda \|\mathbf{w}\|^2$$

In matrix form: $J(\mathbf{w}) = \|\mathbf{y} - \mathbf{\Phi}\mathbf{w}\|^2 + \lambda\|\mathbf{w}\|^2$.

Apply Theorem: We substitute $\mathbf{w} = \sum_j \alpha_j \phi(\mathbf{x}_j) = \mathbf{\Phi}^\top \boldsymbol{\alpha}$.

1. The prediction vector $\mathbf{\Phi}\mathbf{w}$ becomes:

$$\mathbf{\Phi}\mathbf{w} = \mathbf{\Phi}(\mathbf{\Phi}^\top \boldsymbol{\alpha}) = (\mathbf{\Phi}\mathbf{\Phi}^\top)\boldsymbol{\alpha} = \mathbf{K}\boldsymbol{\alpha}$$

   where $\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}^\top$ is the $N \times N$ Gram matrix ($K_{ij} = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$).

2. The regularizer $\|\mathbf{w}\|^2$ becomes:

$$\|\mathbf{w}\|^2 = \mathbf{w}^\top \mathbf{w} = (\mathbf{\Phi}^\top \boldsymbol{\alpha})^\top (\mathbf{\Phi}^\top \boldsymbol{\alpha}) = \boldsymbol{\alpha}^\top (\mathbf{\Phi}\mathbf{\Phi}^\top)\boldsymbol{\alpha} = \boldsymbol{\alpha}^\top \mathbf{K}\boldsymbol{\alpha}$$

Kernelized Objective: By substituting these into $J(\mathbf{w})$, we get the kernelized objective $J(\boldsymbol{\alpha})$:

$$J(\boldsymbol{\alpha}) = \|\mathbf{y} - \mathbf{K}\boldsymbol{\alpha}\|^2 + \lambda \boldsymbol{\alpha}^\top \mathbf{K}\boldsymbol{\alpha}$$

This is a convex quadratic function of $\boldsymbol{\alpha}$. We solve by setting $\nabla_{\boldsymbol{\alpha}} J = 0$:

$$\nabla_{\boldsymbol{\alpha}} J = \nabla_{\boldsymbol{\alpha}} (\mathbf{y}^\top \mathbf{y} - 2\mathbf{y}^\top \mathbf{K}\boldsymbol{\alpha} + \boldsymbol{\alpha}^\top \mathbf{K}^2 \boldsymbol{\alpha}) + \lambda(2\mathbf{K}\boldsymbol{\alpha})$$

$$\nabla_{\boldsymbol{\alpha}} J = -2\mathbf{K}^\top \mathbf{y} + 2\mathbf{K}^\top \mathbf{K}\boldsymbol{\alpha} + 2\lambda \mathbf{K}\boldsymbol{\alpha} = 0$$

Since $\mathbf{K}^\top = \mathbf{K}$:

$$-2\mathbf{K}\mathbf{y} + 2\mathbf{K}\mathbf{K}\boldsymbol{\alpha} + 2\lambda \mathbf{K}\boldsymbol{\alpha} = 0$$

$$\mathbf{K}(\mathbf{K}\boldsymbol{\alpha} - \mathbf{y} + \lambda \mathbf{I}\boldsymbol{\alpha}) = 0$$

This implies the solution satisfies:

$$(\mathbf{K} + \lambda \mathbf{I})\boldsymbol{\alpha} = \mathbf{y}$$

KRR Solution

The coefficients are $\boldsymbol{\alpha}^\star = (\mathbf{K} + \lambda \mathbf{I})^{-1}\mathbf{y}$. The prediction for a new point $\mathbf{x}$ is:

$$f(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) = (\mathbf{\Phi}^\top \boldsymbol{\alpha})^\top \phi(\mathbf{x}) = \boldsymbol{\alpha}^\top \mathbf{\Phi}\phi(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i K(\mathbf{x}_i, \mathbf{x})$$

2. Kernel Logistic Regression

Primal Problem: We minimize the L2-regularized negative log-likelihood (binary cross-entropy). Let $y_i \in \{-1, +1\}$.

$$J(\mathbf{w}) = \sum_{i=1}^{N} \log\big(1 + \exp\big(-y_i \mathbf{w}^\top \phi(\mathbf{x}_i)\big)\big) + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

Apply Theorem: Substitute $\mathbf{w} = \sum_j \alpha_j \phi(\mathbf{x}_j)$.

1. The score $\mathbf{w}^\top \phi(\mathbf{x}_i)$ becomes:

$$\mathbf{w}^\top \phi(\mathbf{x}_i) = \left(\sum_j \alpha_j \phi(\mathbf{x}_j) \right)^\top \phi(\mathbf{x}_i) = \sum_j \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{K}\boldsymbol{\alpha})_i$$

   This is the i-th element of the vector $\mathbf{K}\boldsymbol{\alpha}$.

2. The regularizer $\|\mathbf{w}\|^2$ becomes $\boldsymbol{\alpha}^\top \mathbf{K}\boldsymbol{\alpha}$.

Kernelized Objective:

Kernel Logistic Regression Objective

$$J(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \log(1 + \exp(-y_i (\mathbf{K}\boldsymbol{\alpha})_i)) + \frac{\lambda}{2}\boldsymbol{\alpha}^\top \mathbf{K}\boldsymbol{\alpha}$$

There is **no closed-form solution** for $\boldsymbol{\alpha}$. This N-dimensional problem must be solved numerically (e.g., using gradient descent or Newton's method on $\boldsymbol{\alpha}$).

3. Support Vector Machine (SVM)

Primal Problem: The SVM primal objective (in hinge-loss form) is:

$$J(\mathbf{w}, b) = \sum_{i=1}^{N} \max(0, 1 - y_i(\mathbf{w}^\top \phi(\mathbf{x}_i) + b)) + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

(This is the "λ-SVM" form, which is equivalent to the "C-SVM" form where $C \propto 1/\lambda$).

Apply Theorem: The Representer Theorem guarantees the solution $\mathbf{w}^\star$ must lie in the span of the data:

$$\mathbf{w}^\star = \sum_{j=1}^{N} \beta_j \phi(\mathbf{x}_j)$$

The standard SVM dual derivation is a more direct way to find the coefficients. It shows that they have a specific form: $\beta_j = \alpha_j y_j$, where α_j are the dual variables.

$$\mathbf{w}^\star = \sum_{j=1}^{N} \alpha_j y_j \phi(\mathbf{x}_j)$$

This $\mathbf{w}$ is what we derived from the KKT stationarity condition earlier. The Representer Theorem is the underlying justification for why $\mathbf{w}$ *must* take this form.

Kernelized Prediction Function: We kernelize the prediction function $f(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + b$ by substituting this form of $\mathbf{w}$:

$$f(\mathbf{x}) = \left(\sum_{i=1}^{N} \alpha_i y_i \phi(\mathbf{x}_i) \right)^\top \phi(\mathbf{x}) + b$$

$$= \sum_{i=1}^{N} \alpha_i y_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle + b$$

SVM Kernelized Prediction

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

The coefficients $\boldsymbol{\alpha}$ are found not by minimizing a kernelized primal, but by solving the (simpler) kernelized dual problem (which we've already covered).

Kernelised Ridge Regression (KRR)

1. Primal Problem in Feature Space

Let $\phi : \mathbb{R}^D \to \mathcal{H}$ be a feature map. The primal problem for KRR minimizes the L2-regularized squared loss in the feature space $\mathcal{H}$.

$$\min_{\mathbf{w}\in\mathcal{H}} \frac{1}{2}\sum_{i=1}^{N}(y_i - \mathbf{w}^\top\phi(\mathbf{x}_i))^2 + \frac{\lambda}{2}\|\mathbf{w}\|_{\mathcal{H}}^2$$

(We omit the bias term b for simplicity, assuming data is centered or b is absorbed via feature augmentation).
Let $\boldsymbol{\Phi}$ be the $N \times \dim(\mathcal{H})$ matrix whose i-th row is $\phi(\mathbf{x}_i)^\top$. The objective in matrix form is:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{y} - \boldsymbol{\Phi}\mathbf{w}\|^2 + \frac{\lambda}{2}\|\mathbf{w}\|_{\mathcal{H}}^2$$

Taking the derivative w.r.t. $\mathbf{w}$ and setting to zero gives the normal equations in feature space:

$$\boldsymbol{\Phi}^\top(\boldsymbol{\Phi}\mathbf{w} - \mathbf{y}) + \lambda\mathbf{w} = \mathbf{0} \quad\Longrightarrow\quad (\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \lambda\mathbf{I})\mathbf{w} = \boldsymbol{\Phi}^\top\mathbf{y} \tag{P1}$$

2. Dual Form Derivation

We use the Representer Theorem, which states the solution $\mathbf{w}$ must lie in the span of the training features: $\mathbf{w} = \boldsymbol{\Phi}^\top\boldsymbol{\alpha}$ for some $\boldsymbol{\alpha} \in \mathbb{R}^N$. We substitute this into the primal objective $J(\mathbf{w})$:

- **Residual Term:** $\mathbf{y} - \boldsymbol{\Phi}\mathbf{w} = \mathbf{y} - \boldsymbol{\Phi}(\boldsymbol{\Phi}^\top\boldsymbol{\alpha}) = \mathbf{y} - (\boldsymbol{\Phi}\boldsymbol{\Phi}^\top)\boldsymbol{\alpha} = \mathbf{y} - \mathbf{K}\boldsymbol{\alpha}$
- **Regularizer Term:** $\|\mathbf{w}\|_{\mathcal{H}}^2 = \|\boldsymbol{\Phi}^\top\boldsymbol{\alpha}\|_{\mathcal{H}}^2 = (\boldsymbol{\Phi}^\top\boldsymbol{\alpha})^\top(\boldsymbol{\Phi}^\top\boldsymbol{\alpha}) = \boldsymbol{\alpha}^\top(\boldsymbol{\Phi}\boldsymbol{\Phi}^\top)\boldsymbol{\alpha} = \boldsymbol{\alpha}^\top\mathbf{K}\boldsymbol{\alpha}$

Here, $\mathbf{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^\top$ is the $N \times N$ kernel (Gram) matrix. The objective, now a function of $\boldsymbol{\alpha}$, is:

$$J(\boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{y} - \mathbf{K}\boldsymbol{\alpha}\|^2 + \frac{\lambda}{2}\boldsymbol{\alpha}^\top\mathbf{K}\boldsymbol{\alpha}$$

Differentiate w.r.t. $\boldsymbol{\alpha}$ and set to zero:

$$\nabla_{\boldsymbol{\alpha}} J = -\mathbf{K}^\top(\mathbf{y} - \mathbf{K}\boldsymbol{\alpha}) + \lambda\mathbf{K}\boldsymbol{\alpha} = \mathbf{0}$$

Since $\mathbf{K}$ is symmetric ($\mathbf{K}^\top = \mathbf{K}$):

$$-\mathbf{K}\mathbf{y} + \mathbf{K}^2\boldsymbol{\alpha} + \lambda\mathbf{K}\boldsymbol{\alpha} = \mathbf{0}$$

$$\mathbf{K}(\mathbf{K} + \lambda\mathbf{I})\boldsymbol{\alpha} = \mathbf{K}\mathbf{y}$$

Assuming $\mathbf{K}$ is invertible (or restricting to its column space), we get:

$$(\mathbf{K} + \lambda\mathbf{I})\boldsymbol{\alpha} = \mathbf{y}$$

$$\boxed{\boldsymbol{\alpha} = (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y}}$$

3. Prediction Formula

Given $\boldsymbol{\alpha}$, the prediction for a new point $\mathbf{x}$ is:

$$\begin{aligned}\hat{f}(\mathbf{x}) &= \mathbf{w}^\top\phi(\mathbf{x}) = (\boldsymbol{\Phi}^\top\boldsymbol{\alpha})^\top\phi(\mathbf{x}) = \boldsymbol{\alpha}^\top\boldsymbol{\Phi}\phi(\mathbf{x}) \\ &= \sum_{i=1}^{N}\alpha_i\langle\phi(\mathbf{x}_i), \phi(\mathbf{x})\rangle = \sum_{i=1}^{N}\alpha_i K(\mathbf{x}_i, \mathbf{x})\end{aligned}$$

In vector form, $k(\mathbf{x}, \mathbf{X}) = [K(\mathbf{x}, \mathbf{x}_1), \ldots, K(\mathbf{x}, \mathbf{x}_N)]$:

$$\boxed{f(\mathbf{x}) = k(\mathbf{x}, \mathbf{X})\boldsymbol{\alpha}}$$

The vector of fitted values on the training set is $\hat{\mathbf{y}} = \mathbf{K}\boldsymbol{\alpha}$.

4. Primal-Dual Equivalence

The explicit link is $\mathbf{w} = \boldsymbol{\Phi}^\top\boldsymbol{\alpha}$. Substituting the solution for $\boldsymbol{\alpha}$:

$$\boxed{\mathbf{w} = \boldsymbol{\Phi}^\top\boldsymbol{\alpha} = \boldsymbol{\Phi}^\top(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y}}$$

$\mathbf{w}$ (primal) is a linear combination of training feature vectors $\phi(\mathbf{x}_i)$ with coefficients α_i (dual).

5. RKHS Norm as Regularizer

The primal regularizer $\|\mathbf{w}\|_{\mathcal{H}}^2$ is equivalent to a penalty on $\boldsymbol{\alpha}$:

$$\|\mathbf{w}\|_{\mathcal{H}}^2 = \|\boldsymbol{\Phi}^\top\boldsymbol{\alpha}\|_{\mathcal{H}}^2 = \boldsymbol{\alpha}^\top(\boldsymbol{\Phi}\boldsymbol{\Phi}^\top)\boldsymbol{\alpha} = \boldsymbol{\alpha}^\top\mathbf{K}\boldsymbol{\alpha}$$

Minimizing $\|\mathbf{w}\|^2$ is exactly penalizing the RKHS norm of the predictor.

6. Why λ Ensures Invertibility

$\mathbf{K}$ is symmetric positive semidefinite (PSD). Its eigendecomposition is $\mathbf{K} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$, where $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$ with $\lambda_j \geq 0$.

$$\mathbf{K} + \lambda\mathbf{I} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top + \lambda\mathbf{U}\mathbf{I}\mathbf{U}^\top = \mathbf{U}(\boldsymbol{\Lambda} + \lambda\mathbf{I})\mathbf{U}^\top$$

The eigenvalues of $\mathbf{K} + \lambda\mathbf{I}$ are $(\lambda_j + \lambda)$. Since $\lambda > 0$ and $\lambda_j \geq 0$, every eigenvalue $(\lambda_j + \lambda)$ is strictly positive. A symmetric matrix with strictly positive eigenvalues is positive definite (PD) and thus invertible. This is Tikhonov regularization.

7. LOOCV Formula for KRR

KRR is a linear smoother: $\hat{\mathbf{y}} = \mathbf{H}\mathbf{y}$, with hat matrix $\mathbf{H} = \mathbf{K}(\mathbf{K} + \lambda\mathbf{I})^{-1}$. The leave-one-out (LOO) residual for point i can be computed cheaply without re-training:

$$y_i - \hat{y}_i^{(-i)} = \frac{y_i - \hat{y}_i}{1 - H_{ii}}$$

The LOOCV Mean Squared Error is:

$$\text{LOOCV} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{y_i - \hat{y}_i}{1 - H_{ii}}\right)^2$$

This avoids recomputing N models.

8. Computational Cost

- **Training (Gram Matrix):** $O(N^2 D)$ to compute $\mathbf{K}$.
- **Training (Solve):** $O(N^3)$ to solve $(\mathbf{K} + \lambda\mathbf{I})\boldsymbol{\alpha} = \mathbf{y}$ via Cholesky decomposition.
- **Prediction:** $O(ND)$ to compute $k(\mathbf{x}, \mathbf{X})$ and $O(N)$ for the final dot product. Total: $O(ND)$.

The $O(N^3)$ training cost limits direct KRR to $N \lesssim 10^4$.

9. KRR vs. Kernel OLS

Kernel OLS would solve $\mathbf{K}\boldsymbol{\alpha} = \mathbf{y}$. This is bad:

1. $\mathbf{K}$ is often singular (e.g., if $D > N$) or ill-conditioned.
2. $\lambda > 0$ ensures $\mathbf{K} + \lambda\mathbf{I}$ is invertible and stable.
3. λ controls the bias-variance tradeoff and prevents overfitting.

10. Toy Numeric Example (3-point dataset)

Let $\mathbf{x} = [1, 2, 3]^\top$, $\mathbf{y} = [1, 2, 3]^\top$. Use linear kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i\mathbf{x}_j$ and $\lambda = 1$.

$$\mathbf{K} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{bmatrix}, \quad \lambda\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$(\mathbf{K} + \lambda\mathbf{I}) = \begin{bmatrix} 2 & 2 & 3 \\ 2 & 5 & 6 \\ 3 & 6 & 10 \end{bmatrix}$$

Solve $(\mathbf{K} + \lambda\mathbf{I})\boldsymbol{\alpha} = \mathbf{y}$:

$$\begin{pmatrix} 2 & 2 & 3 \\ 2 & 5 & 6 \\ 3 & 6 & 10 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \quad \Longrightarrow \quad \boldsymbol{\alpha} = \begin{pmatrix} 1/15 \\ 2/15 \\ 3/15 \end{pmatrix}$$

Predictions on training points:

$$\hat{\mathbf{y}} = \mathbf{K}\boldsymbol{\alpha} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{bmatrix} \begin{pmatrix} 1/15 \\ 2/15 \\ 3/15 \end{pmatrix} = \begin{pmatrix} 14/15 \\ 28/15 \\ 42/15 \end{pmatrix} \approx \begin{pmatrix} 0.933 \\ 1.867 \\ 2.800 \end{pmatrix}$$

The predictions are shrunk towards zero by the regularizer.

11. Why λ Controls Smoothness

As $\lambda \to \infty$, the solution $\boldsymbol{\alpha} = (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y} \to \mathbf{0}$. This makes $f(\mathbf{x}) \to 0$. As $\lambda \to 0$, $\boldsymbol{\alpha} \to \mathbf{K}^{-1}\mathbf{y}$, which is the non-regularized (and likely overfit) solution. Spectrally, $\boldsymbol{\alpha} = \mathbf{U}(\boldsymbol{\Lambda} + \lambda\mathbf{I})^{-1}\mathbf{U}^\top\mathbf{y}$. A large λ heavily dampens components corresponding to small eigenvalues λ_j, which often represent high-frequency/non-smooth functions. Thus, increasing λ increases smoothness.

Final Compact Summary

- **Dual solution:** $\boldsymbol{\alpha} = (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y}$.
- **Prediction:** $f(\mathbf{x}) = k(\mathbf{x}, \mathbf{X})\boldsymbol{\alpha} = \sum_i \alpha_i K(\mathbf{x}_i, \mathbf{x})$.
- **Primal-dual link:** $\mathbf{w} = \boldsymbol{\Phi}^\top\boldsymbol{\alpha}$ and $\|\mathbf{w}\|^2 = \boldsymbol{\alpha}^\top\mathbf{K}\boldsymbol{\alpha}$ (RKHS norm).
- **Invertibility:** $\lambda > 0 \implies (\mathbf{K} + \lambda\mathbf{I})$ is PD and invertible.
- **LOOCV:** Use hat matrix $\mathbf{H} = \mathbf{K}(\mathbf{K} + \lambda\mathbf{I})^{-1}$ and LOO residual $\frac{y_i - \hat{y}_i}{1 - H_{ii}}$.

Kernelised SVM

1. How the SVM Dual is Kernelized

We start from the soft-margin dual, expressed in a general feature space $\mathcal{H}$ using the map ϕ:

$$\max_{\boldsymbol{\alpha}} \quad L_D(\boldsymbol{\alpha}) = \sum_{i=1}^N \alpha_i - \frac{1}{2}\sum_{i=1}^N\sum_{j=1}^N \alpha_i\alpha_j y_i y_j \langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle_{\mathcal{H}}$$

$$\text{s.t.} \quad 0 \le \alpha_i \le C, \quad \sum_{i=1}^N \alpha_i y_i = 0.$$

The **kernel trick** is to replace every inner product $\langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle$ with a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$. This gives the final, solvable **Kernelized SVM Dual**:

Kernelized SVM Dual Problem

$$\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^N \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{s.t.} \quad 0 \le \alpha_i \le C, \quad \sum_{i=1}^N \alpha_i y_i = 0.$$

The problem is now entirely in terms of the $N \times N$ Gram matrix $\mathbf{K}$ and the coefficients $\boldsymbol{\alpha}$, with no explicit reference to the (possibly infinite-dimensional) feature space.

2. Kernel SVM Decision Function

From the KKT stationarity condition, the optimal primal weight vector $\mathbf{w}^\star$ (in feature space $\mathcal{H}$) is:

$$\mathbf{w}^\star = \sum_{i=1}^N \alpha_i^\star y_i \phi(\mathbf{x}_i)$$

The prediction for a new test point $\mathbf{x}$ is $f(\mathbf{x}) = (\mathbf{w}^\star)^\top\phi(\mathbf{x}) + b^\star$. Substituting $\mathbf{w}^\star$:

$$f(\mathbf{x}) = \left(\sum_{i=1}^N \alpha_i^\star y_i \phi(\mathbf{x}_i)\right)^\top \phi(\mathbf{x}) + b^\star$$

$$= \sum_{i=1}^N \alpha_i^\star y_i \langle\phi(\mathbf{x}_i), \phi(\mathbf{x})\rangle + b^\star$$

Applying the kernel trick $\langle\phi(\mathbf{x}_i), \phi(\mathbf{x})\rangle = K(\mathbf{x}_i, \mathbf{x})$:

Kernel SVM Decision Function

$$f(\mathbf{x}) = \sum_{i=1}^N \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

The prediction is $\hat{y} = \mathrm{sign}(f(\mathbf{x}))$. The sum only includes support vectors (where $\alpha_i > 0$).

3. Why Support Vectors Depend on Kernel Choice

Support vectors are the training indices i with $\alpha_i > 0$ at the optimum.

- The kernel K defines the quadratic part of the dual objective via the matrix $\mathbf{Q}$, where $Q_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$.
- Changing K changes $\mathbf{Q}$, which fundamentally changes the geometry of the optimization problem (its curvature and the coupling between variables).
- Therefore, the optimal solution $\boldsymbol{\alpha}^\star$ depends on K.
- **Geometrically:** The kernel defines the feature map ϕ and thus the geometry (distances and angles) of the data in feature space. A point $\mathbf{x}_i$ that is a support vector (on or inside the margin) for one kernel may be far from the margin (and thus $\alpha_i = 0$) for another.

4. Linear vs. Kernel SVM Decision Boundaries

Linear SVM (This is just $K(\mathbf{x}, \mathbf{z}) = \mathbf{x}^\top\mathbf{z}$, so $\phi(\mathbf{x}) = \mathbf{x}$).

- The decision boundary is a hyperplane in the **input space**:

$$f(\mathbf{x}) = \mathbf{w}^\top\mathbf{x} + b = 0$$

- This is a linear separator (a line in 2D, a plane in 3D).

Kernel SVM (non-linear kernel)

- The SVM finds a linear hyperplane in the **feature space** $\mathcal{H}$:

$$\mathbf{w}^\top \phi(\mathbf{x}) + b = 0$$

- When this linear boundary from $\mathcal{H}$ is mapped back to the original input space, it becomes a **nonlinear decision boundary**.
- A polynomial kernel K gives a polynomial boundary. An RBF kernel K gives a flexible, contour-like boundary.

5. RBF SVM can Separate Any Finite Dataset

Let the Gaussian (RBF) kernel be $K(\mathbf{x}, \mathbf{z}) = \exp(-\gamma \|\mathbf{x} - \mathbf{z}\|^2)$.

1. **Key Fact:** For any set of distinct training points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, the Gaussian Gram matrix $\mathbf{K}$ (where $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$) is **strictly positive definite** and thus invertible.
2. **Constructive Proof:** Since $\mathbf{K}$ is invertible, for any target vector $\mathbf{t} \in \mathbb{R}^N$, we can solve $\mathbf{K}\boldsymbol{\alpha} = \mathbf{t}$ to find $\boldsymbol{\alpha} = \mathbf{K}^{-1}\mathbf{t}$.
3. Let's choose our target vector to be the (scaled) labels: $\mathbf{t} = \tau\mathbf{y}$, where $\mathbf{y} \in \{-1, +1\}^N$ are the labels and $\tau > 0$ is a margin.
4. This gives $\boldsymbol{\alpha} = \mathbf{K}^{-1}(\tau\mathbf{y})$.
5. Consider the kernel SVM decision function $f(\mathbf{x}) = \sum_j \alpha_j K(\mathbf{x}_j, \mathbf{x})$ (with $b = 0$). Let's evaluate this function on the training points:

$$f(\mathbf{x}_i) = \sum_{j=1}^{N} \alpha_j K(\mathbf{x}_j, \mathbf{x}_i) = (\mathbf{K}\boldsymbol{\alpha})_i = (\mathbf{t})_i = \tau y_i$$

6. Check the margin: $y_i f(\mathbf{x}_i) = y_i(\tau y_i) = \tau y_i^2 = \tau > 0$.

This shows that a function of the form $f(\mathbf{x}) = \sum_i \alpha_i K(\mathbf{x}_i, \mathbf{x})$ exists that perfectly separates any finite, distinct dataset with a margin of τ. An SVM with a sufficiently large C will find such a separating hyperplane.

1. Effect of Extremely Large Kernel Bandwidth ($\sigma \to \infty$)

For the RBF kernel, $K(\mathbf{x}, \mathbf{z}) = \exp(-\|\mathbf{x} - \mathbf{z}\|^2 / (2\sigma^2))$. As $\sigma \to \infty$, the exponent approaches zero:

$$\frac{\|\mathbf{x} - \mathbf{z}\|^2}{2\sigma^2} \to 0 \quad \Rightarrow \quad K(\mathbf{x}, \mathbf{z}) \to e^0 = 1$$

The kernel matrix $\mathbf{K}$ becomes a matrix of all ones:

$$K(\mathbf{x}, \mathbf{z}) \approx 1 \quad \forall \mathbf{x}, \mathbf{z} \quad \Rightarrow \quad \mathbf{K} \approx \mathbf{1}\mathbf{1}^\top$$

Consequences:

- All points are treated as equally similar (identical) in the feature space.
- The Gram matrix $\mathbf{K}$ becomes a rank-1 matrix.
- The SVM sees no structure in the data and loses all nonlinearity. The decision boundary collapses to a single linear classifier.
- The model **heavily underfits**.

2. Effect of Extremely Small Kernel Bandwidth ($\sigma \to 0$)

As $\sigma \to 0$, the exponent approaches infinity for any non-identical points.

- If $\mathbf{x} = \mathbf{z}$: $K(\mathbf{x}, \mathbf{x}) = \exp(0) = 1$.
- If $\mathbf{x} \neq \mathbf{z}$: $K(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{z}\|^2}{2\sigma^2}\right) \to 0$.

The kernel matrix $\mathbf{K}$ becomes the identity matrix:

$$\mathbf{K} \approx \mathbf{I}_N$$

Consequences:

- Every point is orthogonal to every other point in the feature space.
- The feature space is extremely "spread-out," and any finite dataset becomes linearly separable.
- An SVM with a large C will fit every single training point exactly, leading to **severe overfitting**.
- The classifier "memorizes" the training data and behaves like a nearest-neighbor classifier.

3. Kernel SVM Overfitting Behaviour

SVM overfits when the classifier becomes too flexible and "memorizes" the training data. This is caused by a combination of:

- **Small bandwidth (σ):** As shown above, this makes the kernel too localized and "spiky," allowing the model to draw complex boundaries that isolate individual points.
- **Large C (low regularization):** The objective is $\min \frac{1}{2}\|\mathbf{w}\|^2 + C \sum \xi_i$. A large C heavily penalizes any slack ξ_i, forcing the model to achieve zero training error (or as close as possible), even if it requires a very complex boundary (a large $\|\mathbf{w}\|$ in the feature space).

$$\textbf{(small bandwidth } \sigma\textbf{)} + \textbf{(large } C\textbf{)} \quad \Rightarrow \quad \text{Overfitting}$$

4. Role of Kernel Matrix Conditioning

The $N \times N$ Gram matrix $\mathbf{K}$ plays the same role for the dual problem as the covariance matrix $\mathbf{X}^\top\mathbf{X}$ does for the primal.

- **Well-conditioned $\mathbf{K}$:** All eigenvalues are reasonably large and not too close to zero. The dual optimization is stable. The solution (e.g., in KRR, $\boldsymbol{\alpha} = (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y}$) is stable and behaves smoothly. This leads to good generalization.
- **Ill-conditioned $\mathbf{K}$ (nearly singular):** The matrix has very small eigenvalues. This can happen if σ is too large (matrix → rank-1) or if data points are duplicated or nearly collinear in the feature space.
  - This leads to numerical instability in the QP solver.
  - The resulting $\boldsymbol{\alpha}$ vector can be huge and highly sensitive to small changes in the data, leading to poor generalization.

5. How Kernel Centering Affects SVM

Centering the kernel corresponds to centering the data in the feature space $\mathcal{H}$. Given $\mathbf{K}$, the centered kernel matrix is $\mathbf{K}_c = \mathbf{H}\mathbf{K}\mathbf{H}$, where $\mathbf{H} = \mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^\top$ is the centering matrix. This is equivalent to using the centered feature map:

$$\phi_c(\mathbf{x}) = \phi(\mathbf{x}) - \frac{1}{N}\sum_{j=1}^{N} \phi(\mathbf{x}_j) = \phi(\mathbf{x}) - \bar{\phi}$$

The inner product changes:

$$\langle \phi_c(\mathbf{x}_i), \phi_c(\mathbf{x}_j) \rangle = K_{ij} - \bar{\phi}^\top \phi(\mathbf{x}_i) - \bar{\phi}^\top \phi(\mathbf{x}_j) + \|\bar{\phi}\|^2$$

This changes the geometry of the feature space and thus the resulting hyperplane.

However: The standard SVM model includes a bias term b. The SVM is translation-invariant in the feature space. A model trained on centered data, $(\mathbf{w}_c)^\top \phi_c(\mathbf{x}) + b_c$, is equivalent to a model trained on uncentered data, $\mathbf{w}^\top \phi(\mathbf{x}) + b$.

$$\mathbf{w}^\top(\phi(\mathbf{x}) - \bar{\phi}) + b' = \mathbf{w}^\top \phi(\mathbf{x}) + (b' - \mathbf{w}^\top \bar{\phi})$$

The model simply learns a different bias term $b = (b' - \mathbf{w}^\top \bar{\phi})$ and the same $\mathbf{w}$.

Practical Summary:

- For SVM **with a bias term** b, centering the kernel has no effect on the final decision boundary.
- For SVM **without a bias term** b, centering matters and will change the solution.
- (Note: Kernel PCA always requires a centered kernel.)

Final Summary

- **Bandwidth Effects:**
  - $\sigma \to \infty$: $K \to \mathbf{1}\mathbf{1}^\top$ (rank 1) $\implies$ **Underfits** (linear classifier).
  - $\sigma \to 0$: $K \to \mathbf{I}$ (identity) $\implies$ **Overfits** (memorizes data).
- **SVM Overfitting:** Caused by **small** σ (high flexibility) + **large** C (low regularization).
- **Conditioning:** An ill-conditioned $\mathbf{K}$ (nearly singular) leads to an unstable optimization and poor generalization.
- **Centering:** Centers the feature map $\phi_c(\mathbf{x}) = \phi(\mathbf{x}) - \bar{\phi}$. This **does not affect** the final SVM decision boundary as long as a bias term b is included in the model.