ML Cheat Sheet

Closed-form solutions

Closed-form solutions exist only for simple models with simple loss (typically quadratic) and assumptions. Closed-form solutions examples: Ordinary Least Squares(OLS), Ridge Regression, Weighted Least Squares, Gaussian Maximum Likelihood Estimation(MLE).

Matrix inversion costs $O(n^3)$, which does not scale. If $X^T X$ is singular or ill-conditioned, the solution does not exist or is unstable.

Linear Model

To find the minimum of the loss function, we compute partial derivatives:

$$\frac{\partial \ell(\beta_0, \beta_1)}{\partial \beta_1} = 0, \qquad \frac{\partial \ell(\beta_0, \beta_1)}{\partial \beta_0} = 0.$$

Solving these equations gives:

$$\beta_1 = \frac{\sum x_i y_i - n\bar{x}\bar{y}}{\sum x_i^2 - n\bar{x}^2} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2} = \frac{\mathrm{Cov}(x, y)}{\mathrm{Var}(x)}.$$

The intercept is:

$$\beta_0 = \bar{y} - \beta_1 \bar{x}.$$

Consider a linear model with multiple outputs:

$$y_i = \sum_{j=1}^{n} x_{ij} w_j$$

For a fixed weight w_k:

$$\frac{\partial y_i}{\partial w_k} = \frac{\partial}{\partial w_k} \left(\sum_{j=1}^{n} x_{ij} w_j \right) = x_{ik}.$$

Stacking all derivatives together gives:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{w}} = X^\top.$$

For a function:

$$f : \mathbb{R}^n \to \mathbb{R}^m, \quad f(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{bmatrix},$$

the derivative is the **Jacobian matrix**:

$$J = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}.$$

Each row corresponds to one output, and each column corresponds to one input dimension.

For a scalar-valued function $f : \mathbb{R}^n \to \mathbb{R}$, the **Hessian matrix** captures second-order derivatives:

$$H(f) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}.$$

The Hessian describes **curvature(曲率)**. In optimization:

- Positive definite(正定) Hessian → unique global minimum
- Indefinite(不定) Hessian → saddle points(鞍点) may exist

For squared loss in linear regression, the Hessian turns out to be constant(常数) and positive semidefinite(半正定), which guarantees convexity(凸性).

Rewrite the model in vectorized form:

$$\hat{\mathbf{Y}} = XW,$$

where the first column of X represent the bias (intercept) term. Using squared error, the loss becomes:

$$\ell(W) = ||Y - XW||_F^2.$$

This **compact expression encodes all data points simultaneously.** Taking the derivative with respect to W:

$$\frac{\partial \ell(W)}{\partial W} = -2X^\top (Y - XW),$$

which means **the direction of steepest descent in parameter space**. Setting the gradient to zero yields the **normal equation**:

$$X^\top XW = X^\top Y,$$

which is **to characterize the optimal solution**.

If $X^\top X$ is **invertible**, the optimal solution is:

$$W^* = (X^\top X)^{-1} X^\top Y.$$

This expression is known as the **least-squares solution**, and $(X^\top X)^{-1} X^\top$ is the **pseudoinverse of** X. Key interpretation:

- **The solution projects** Y **onto the column space of** X**.**
- **Learning is equivalent to geometric projection in feature space.**

Figure 1 below shows the three different datasets (A, B, C) and the same linear model used to predict each of these datasets.

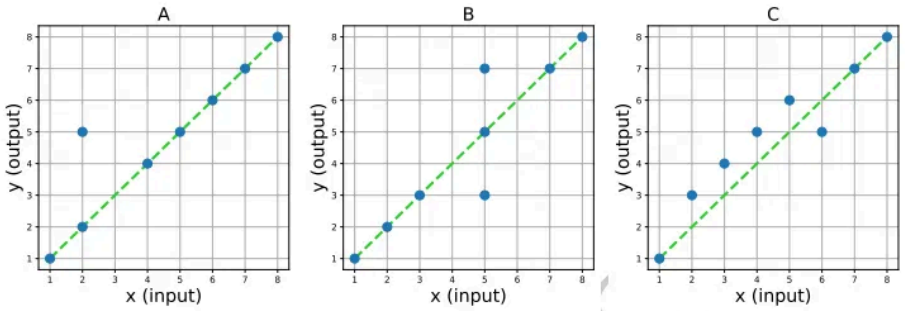


Figure 1: 3 Datasets (A, B, C), and a linear model represented by the dashed line (from $\mathbb{R}^1$ to $\mathbb{R}^1$)

Question 1 Choose the correct ordering with respect to MSE (Mean Squared Error) loss

- ☐ $\text{MSE}_B > \text{MSE}_A > \text{MSE}_C$
- ☐ $\text{MSE}_B > \text{MSE}_C > \text{MSE}_A$
- ☐ At least two of the MSE losses are equal.
- ☐ $\text{MSE}_A > \text{MSE}_C > \text{MSE}_B$
- ☐ $\text{MSE}_C > \text{MSE}_B > \text{MSE}_A$
- ■ $\text{MSE}_A > \text{MSE}_B > \text{MSE}_C$
- ☐ $\text{MSE}_C > \text{MSE}_A > \text{MSE}_B$

Solution: $\text{MSE}_A = 9/8$, $\text{MSE}_B = 8/8$, $\text{MSE}_C = 5/8$

Question 2 Choose the correct ordering with respect to MAE (Mean Absolute Error) loss

- ☐ $\text{MAE}_A > \text{MAE}_C > \text{MAE}_B$
- ☐ $\text{MAE}_B > \text{MAE}_C > \text{MAE}_A$
- ☐ $\text{MAE}_A > \text{MAE}_B > \text{MAE}_C$
- ☐ $\text{MAE}_C > \text{MAE}_A > \text{MAE}_B$
- ☐ $\text{MAE}_B > \text{MAE}_A > \text{MAE}_C$
- ☐ At least two of the MAE losses are equal.
- ■ $\text{MAE}_C > \text{MAE}_B > \text{MAE}_A$

Solution: $\text{MAE}_A = 3/8$, $\text{MAE}_B = 4/8$, $\text{MAE}_C = 5/8$

Likelihood and log-likelihood

For a single observation y, conditioned on x and w, the likelihood is

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

For i.i.d. samples, the joint likelihood is the product over all data points. After we immediately take logs:

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

The first term does not depend on w. Therefore:

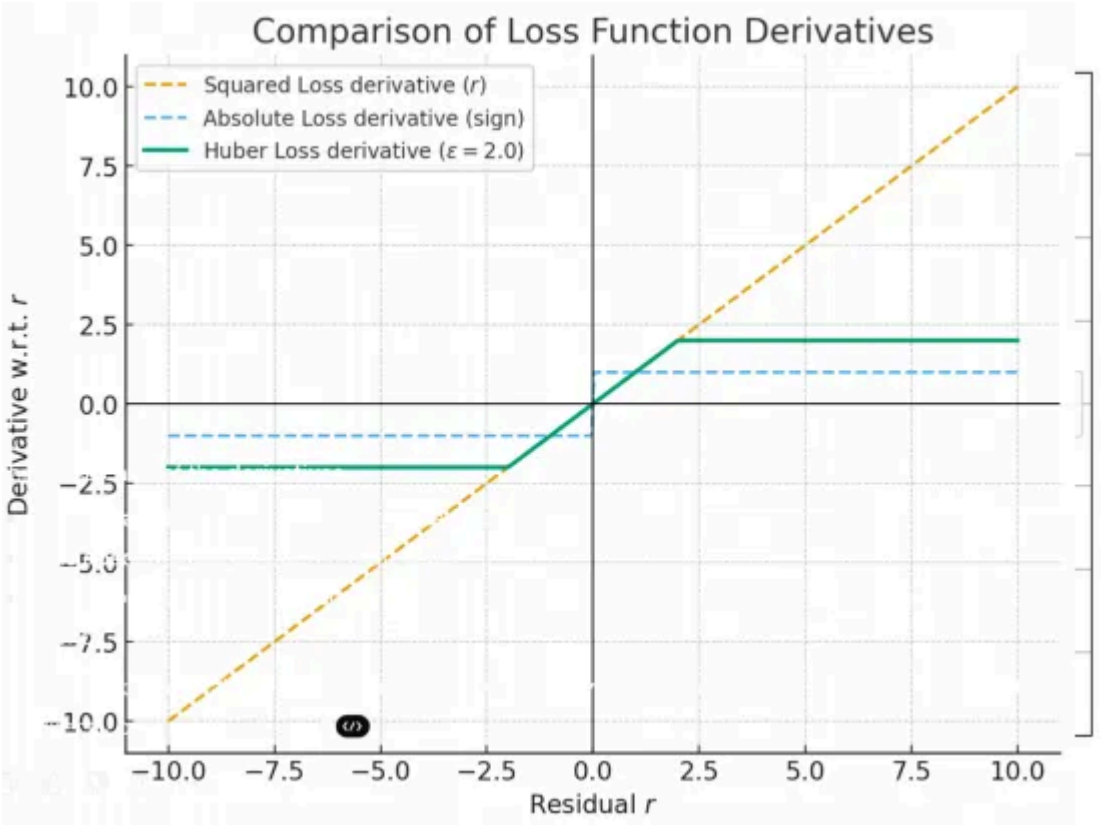
$$\arg\max_w \log p(y \mid X, w) \iff \arg\min_w \sum_{i=1}^{n}(y_i - x_i^\top w)^2.$$

Maximizing the log-likelihood under Gaussian noise is equivalent to minimizing the sum of squared errors.

Choosing a loss function is choosing a statistical assumption. Every loss corresponds to an implicit noise model.

- Gaussian noise ⇒ squared loss
- Laplace noise ⇒ absolute loss / L1 Loss
- Heavy-tailed noise ⇒ robust losses

Huber loss: Bounded like L1 and smooth like L2



Define the residual

$$r_i = f_\beta(x_i) - y_i.$$

The Huber loss with parameter $\varepsilon > 0$ is

$$\ell_\varepsilon(r) = \begin{cases} \frac{1}{2}r^2, & |r| \le \varepsilon \\ \varepsilon\left(|r| - \frac{\varepsilon}{2}\right), & |r| > \varepsilon \end{cases}$$

ε determines the transition point between L2 and L1 behavior.

First compute the derivative of the scalar Huber loss with respect to its argument r:

$$\ell'_\varepsilon(r) = \begin{cases} r, & |r| \le \varepsilon \\ \varepsilon\ \mathrm{sign}(r), & |r| > \varepsilon \end{cases}$$

Now apply the chain rule:

$$\nabla_\beta r_i = x_i.$$

So the gradient of the full objective is

$$\nabla_\beta L(\beta) = \sum_{i=1}^{n} \ell'_\varepsilon(r_i)x_i = \sum_{i:|r_i|\le\varepsilon} r_i x_i + \sum_{i:|r_i|>\varepsilon} \varepsilon\ \mathrm{sign}(r_i)x_i.$$

> Why does the gradient of the Huber objective involve a sum of terms of the form $\ell'_\varepsilon(r_i)x_i$? Because of the chain rule applied to $r_i = x_i^\top \beta - y_i$

In robust statistics language, Huber loss implements **influence capping**.

There is **no closed-form solution** for Huber regression in general. The moment you leave pure L2, you also leave normal equations behind.

> Because the objective is piecewise-defined and not purely quadratic.

Multivariate Linear Regression

Instead of a single feature x, we now consider multiple features:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k.$$

Geometrically, with k features, the model is a hyperplane in $\mathbb{R}^{k+1}$.

For a dataset with n samples and k features, predictions can be written as:

$$\hat{y}_i = \omega_0 + \omega_1 x_{i1} + \cdots + \omega_k x_{ik}, \quad i = 1, \ldots, n.$$

he regression problem is equivalent to solving the linear system:

$$XW = Y.$$

When the number of data points exceeds the number of parameters($n > k$), the system is **overdetermined**. In general:

- There is **no exact solution** satisfying $XW = Y$.
- The system is inconsistent due to noise.

The standard approach is **least squares**, minimizing:

$$||XW - Y||_2^2.$$

If X has full column rank, the solution is:

$$W = (X^\top X)^{-1} X^\top Y.$$

When there are fewer equations than unknowns($n < k$), the system is **underdetermined**. In this case:

- There are **infinitely many exact solutions**.
- $X^\top X$ is singular and not invertible.
- The normal equation formula does not apply.

The **Moore–Penrose pseudoinverse** X^+ provides a unified solution for **any matrix**. It allows us to write:

$$W = X^+ Y$$

as a generalized inverse-based solution. Special cases:

- If $n > k$ and X has full column rank:

$$X^+ = (X^\top X)^{-1} X^\top$$

- If $n < k$ and X has full row rank:

$$X^+ = X^\top (XX^\top)^{-1}$$

For underdetermined systems, the full solution set can be decomposed as:

$$W = W_p + W_h$$

where:

- $W_p = X^\top (XX^\top)^{-1} Y$ is a **particular solution**
- $W_h \in \text{Null}(X)$ is any vector in the null space of X

More explicitly:

$$W = X^\top (XX^\top)^{-1} Y - \left(I - X^\top (XX^\top)^{-1} X\right) z, \quad z \in \mathbb{R}^k$$

Among all possible solutions, the pseudoinverse selects a very special one:

> The solution with **minimum Euclidean norm**.

That is:

$$W^* = \arg\min_W ||W||_2 \quad \text{s.t. } XW = Y.$$

This solution is exactly:

$$W^* = X^+ Y.$$

All other solutions differ by adding null-space components that increase the norm.

Question 1 Assume we have N training samples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ where for each sample $i \in \{1, \ldots, N\}$ we have that $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$. We want to classify the dataset using the exponential loss $L(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} \exp(-y_i \mathbf{x}_i^\top \mathbf{w})$ for $\mathbf{w} \in \mathbb{R}^d$. Which of the following statements is **true**:

- ☐ The loss function L is non-convex in $\mathbf{w}$.
- ☐ There exists a vector $\mathbf{w}^\star$ such that $L(\mathbf{w}^\star) = 0$.
- ■ If I find a vector $\mathbf{w}^\star$ such that $L(\mathbf{w}^\star) < 1/N$, then $\mathbf{w}^*$ linearly separates my dataset.
- ☐ None of the statements are true.
- ☐ This corresponds to doing logistic regression as seen in class.

Solution: $L(w^\star) < 1/N$ implies $\exp(-y_i \mathbf{x}_i \top w^*) < 1 \ \forall i$, which means that $y_i \mathbf{x}_i \top w^\star > 0 \ \forall i$.

即预测值的符号与真实标签一致，所有样本都被正确分类。因此数据是**线性可分**的。

Question 2 Assume we have N training samples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ where for each sample $i \in \{1, \ldots, N\}$ we have that $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. For $\lambda \geq 0$, we consider the following loss:

$$L_\lambda(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \mathbf{x}_i^\top \mathbf{w})^2 + \lambda \|\mathbf{w}\|_2,$$

and let $C_\lambda = \min_{\mathbf{w} \in \mathbb{R}^d} L_\lambda(\mathbf{w})$ denote the optimal loss value. Which of the following statements is **true**:

- ☐ C_λ is a non-increasing function of λ.
- ☐ For $\lambda = 0$, the loss L_0 is convex and has a unique minimizer.
- ☐ None of the statements are true.
- ■ C_λ is a non-decreasing function of λ.

Solution: For $\lambda_1 < \lambda_2$, $L_{\lambda_1}(\mathbf{w}) \leq L_{\lambda_2}(\mathbf{w}) \ \forall \mathbf{w}$, which means that $C_{\lambda_1} \leq C_{\lambda_2}$.

Question 18 Let $\mathbf{x}, \mathbf{w}, \boldsymbol{\delta} \in \mathbb{R}^d$, $y \in \{-1, 1\}$, and $\varepsilon \in \mathbb{R}_{>0}$ be an arbitrary positive value. Which of the following is NOT true in general:

- [x] $\operatorname{argmax}_{\|\boldsymbol{\delta}\|_2 \leq \varepsilon} \log_2(1 + \exp(-y\mathbf{w}^\top(\mathbf{x} + \boldsymbol{\delta}))) = \operatorname{argmax}_{\|\boldsymbol{\delta}\|_2 \leq \varepsilon} \mathbf{1}_{y\mathbf{w}^\top(\mathbf{x}+\boldsymbol{\delta}) \leq 0}$
- [ ] $\operatorname{argmax}_{\|\boldsymbol{\delta}\|_2 \leq \varepsilon} \log_2(1 + \exp(-y\mathbf{w}^\top(\mathbf{x} + \boldsymbol{\delta}))) = \operatorname{argmax}_{\|\boldsymbol{\delta}\|_2 \leq \varepsilon} 1 - \tanh(y\mathbf{w}^\top(\mathbf{x} + \boldsymbol{\delta}))$ where $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
- [ ] $\operatorname{argmax}_{\|\boldsymbol{\delta}\|_2 \leq \varepsilon} \log_2(1 + \exp(-y\mathbf{w}^\top(\mathbf{x} + \boldsymbol{\delta}))) = \operatorname{argmax}_{\|\boldsymbol{\delta}\|_2 \leq \varepsilon} \exp(-y\mathbf{w}^\top(\mathbf{x} + \boldsymbol{\delta}))$
- [ ] $\operatorname{argmax}_{\|\boldsymbol{\delta}\|_2 \leq \varepsilon} \log_2(1 + \exp(-y\mathbf{w}^\top(\mathbf{x} + \boldsymbol{\delta}))) = \operatorname{argmin}_{\|\boldsymbol{\delta}\|_2 \leq \varepsilon} y\mathbf{w}^\top(\mathbf{x} + \boldsymbol{\delta})$

Solution: $\operatorname{argmax}_{\|\boldsymbol{\delta}\|_2 \leq \varepsilon} \log_2(1 + \exp(-\langle \mathbf{w}, \mathbf{x} + \boldsymbol{\delta} \rangle)) = \operatorname{argmax}_{\|\boldsymbol{\delta}\|_2 \leq \varepsilon} \mathbf{1}_{\langle \mathbf{w}, \mathbf{x}+\boldsymbol{\delta} \rangle \leq 0}$ is not true in general since the right-hand side can have multiple maximizers. To correct it, it suffices to substitute $=$ with $\in$. As for the other choices, all the right-hand sides contain losses which are monotonically decreasing in the margin $\mathbf{w}^\top(\mathbf{x} + \boldsymbol{\delta})$ and thus lead to the same maximizer.

Optimization

Bias–variance tradeoff: Increasing model complexity reduces bias but may increase variance.

What is the primary purpose of introducing nonlinear feature mappings $\phi(x)$? To allow nonlinear decision boundaries while keeping optimization linear in parameters

In higher dimensions, **the direction of steepest descent is given by the negative gradient**.

In a contour plot of a loss function, each **contour line** represents points with identical loss values.

Geometrically, the steepest descent direction is always **orthogonal** to the contour lines of the loss function.

Directional derivatives. Let $f(x, y)$ be a scalar-valued function and let $\mathbf{u}$ be a unit vector in the plane We examine how the function value changes if we move a small distance in direction $\mathbf{u}$. Using a first-order Taylor approximation:

$$\Delta z \approx \nabla f(a, b) \cdot \mathbf{u}, \Delta t$$

This shows that **the rate of change of the function** along direction $\mathbf{u}$ depends on the dot product between the gradient and the direction vector.

Formally, the directional derivative of f at point $\mathbf{x}$ in direction $\mathbf{u}$ is defined as:

$$D_{\mathbf{u}} f(\mathbf{x}) = \nabla f(\mathbf{x}) \cdot \mathbf{u}$$

This quantity measures how fast the function increases or decreases when moving along $\mathbf{u}$. If the dot product is positive, the function increases. If it is negative, the function decreases.

Note: The requirement that$\mathbf{u}$ be a unit vector is important because it removes dependence on step size.

Since $\mathbf{u}$ is a unit vector, the directional derivative satisfies:

$$D_{\mathbf{u}}f = ||\nabla f|| \cos(\theta)$$

where θ is the angle between ∇f and $\mathbf{u}$. As $\mathbf{u}$ varies, $\cos(\theta)$ ranges between -1 and 1. Therefore, the maximum possible directional derivative is $||\nabla f||$, achieved when $\mathbf{u}$ points in the same direction as the gradient.

The normalized steepest descent direction is

$$\mathbf{u} = -\frac{\nabla f}{\|\nabla f\|}$$

In practice, this normalization is omitted because the learning rate absorbs the scaling.

Question 16 In matrix factorization for recommender systems, the goal is to approximate a partially observed user-item rating matrix X by decomposing it as the product of two lower-dimensional matrices: W (item features) and Z (user features). The observed ratings x_{dn} represent the rating of the n^{th} user for the d^{th} item, with Ω as the set of indices for observed ratings. Which of the following is true regarding the regularized matrix factorization objective, given by:

$$\min_{W,Z} \frac{1}{2} \sum_{(d,n)\in\Omega} \left(x_{dn} - (WZ^\top)_{dn}\right)^2 + \frac{\lambda_W}{2}\|W\|_F^2 + \frac{\lambda_Z}{2}\|Z\|_F^2$$

- ☐ The optimization requires filling in all missing entries in X before training.
- ☐ The Frobenius norm regularization ensures the sparsity (the number of non-zero entries) of W and Z.
- ☐ The cost function is jointly convex in W and Z.
- ■ Regularization terms $\frac{\lambda_W}{2}\|W\|_F^2$ and $\frac{\lambda_Z}{2}\|Z\|_F^2$ are used to prevent overfitting to the observed ratings.

这是一个**推荐系统里的矩阵分解问题**。有一个用户–物品评分矩阵 X，但**只有一部分评分是已知的，**Ω 表示"哪些位置的评分是观测到的"，想用两个低维矩阵去近似它：物品特征矩阵 W 和用户特征矩阵 Z,用 $WZ^\top$ 去拟合 X

选项 1：❌ **错误 损失函数只对 Ω 中的观测项求和**。没出现的评分，**根本不进目标函数**。优化时不需要、也不应该先填补缺失值。

选项 2：❌**错误** Frobenius 范数的作用是：**让参数变小但不鼓励变成 0。**

选项 3：❌ **错误** 固定 Z，关于 W 是凸的，固定 W，关于 Z 是凸的，**同时对 W, Z 优化时不是凸的。**

Question 4 Consider the loss function $L : \mathbb{R}^d \to \mathbb{R}$, $L(\mathbf{w}) = \frac{\beta}{2}\|\mathbf{w}\|^2$, where $\beta > 0$ is a constant. We run gradient descent on L with a stepsize $\gamma > 0$ starting from some $\mathbf{w}_0 \neq 0$. Which of the statements below is true?

- ☐ Gradient descent converges in two steps for $\gamma = \frac{1}{\beta}$ (i.e., $\mathbf{w}_2$ is the **first** iterate attaining the global minimum of L).
- ☐ Gradient descent with stepsize $\gamma = \frac{2}{\beta}$ produces iterates that diverge to infinity ($\|\mathbf{w}_t\| \to \infty$ as $t \to \infty$).
- ☐ Gradient descent converges to the global minimum for any stepsize $\gamma > 0$.
- ■ Gradient descent converges to the global minimum for any stepsize in the interval $\gamma \in \left(0, \frac{2}{\beta}\right)$.

Solution: The update rule is $\mathbf{w}_{t+1} = \mathbf{w}_t - \gamma\beta\mathbf{w}_t = (1-\gamma\beta)\mathbf{w}_t$. Therefore we have that the sequence $\{\|\mathbf{w}_t\|\}_t$ is given by $\|\mathbf{w}_{t+1}\| = |1-\gamma\beta|\|\mathbf{w}_t\| = |1-\gamma\beta|^t\|\mathbf{w}_0\|$. We can see that for $\gamma = \frac{2}{\beta}$ the elements of the aforementioned sequence never move from $\|\mathbf{w}_0\|$ (so the algorithm does not diverge to infinity for this stepsize). For $\gamma = \frac{1}{\beta}$ the algorithm converges in one step, not two. And finally, for any $\gamma \in \left(0, \frac{2}{\beta}\right)$ the algorithm will converge to the global minimum since $|1-\gamma\beta| \in (0,1)$.

即必须满足 $|1-\gamma\beta| < 1$。

Convergence

We consider linear regression with mean squared error loss:

$$J(w) = \frac{1}{2m}(Xw - y)^T(Xw - y)$$

This objective is a **convex quadratic function**. The gradient is:

$$\nabla J(w) = \frac{1}{m}X^T(Xw - y)$$

Gradient descent updates the parameters as:

$$w_{k+1} = w_k - \alpha \nabla J(w_k)$$

The global minimizer w^* satisfies:

$$\nabla J(w^*) = 0 \quad \Rightarrow \quad w^* = (X^TX)^{-1}X^Ty$$

Define the error vector:

$$e_k = w_k - w^*$$

Substituting(代入) the update rule and using $X^Ty = X^TXw^*$, we obtain:

$$e_{k+1} = \left(I - \frac{\alpha}{m}X^TX\right)e_k$$

This shows that gradient descent behaves like a **linear dynamical system** in the error space. Convergence depends entirely on the **spectral properties** of the matrix:

$$I - \frac{\alpha}{m}X^TX$$

Using the **spectral norm**:

$$||A||_2 = \sup_{z \neq 0} \frac{||Az||_2}{||z||_2}$$

which is equal to **the largest singular value of A**. We obtain the bound:

$$||e_{k+1}||_2 \leq \left|\left|I - \frac{\alpha}{m}X^TX\right|\right|_2 \leq ||e_k||_2$$

For convergence, we require:

$$|1 - \alpha\lambda_i| < 1 \quad \text{for all eigenvalues } \lambda_i \text{ of } \frac{1}{m}X^T X$$

which ensures **contraction of the error mapping**. This yields the fundamental step-size condition:

$$0 < \alpha < \frac{2}{\lambda_{\max}}$$

This is not a heuristic—it is a **necessary and sufficient condition** for convergence in this quadratic case.

Poor conditioning of $X^T X$ causes slow convergence because level sets become highly elongated.

In ill-conditioned problems, tuning the learning rate alone often fails because different directions require incompatible step sizes.

Given a descent direction $-\nabla f(w_t)$, the **Armijo–Goldstein condition** requires:

$$J(w_t - \alpha\nabla f(w_t)) \leq J(w_t) - \frac{1}{2}\alpha||\nabla f(w_t)||^2$$

Interpretation: The actual decrease in loss must be at least a fixed fraction of the predicted first-order decrease. The condition guarantees sufficient decrease relative to first-order prediction.

In practice: Start with a candidate α. **If the condition fails, reduce** α (often by halving). Repeat until the condition is satisfied. This guarantees stability without knowing the Hessian.

Gradient descent uses only first-order information. **Newton's method** exploits **second-order structure**. Using a second-order Taylor expansion around x_t:

$$Q_t(x) = f(x_t) + \nabla f(x_t)^T(x - x_t) + \frac{1}{2}(x - x_t)^T H(x_t)(x - x_t)$$

Here, $H(x_t)$ is the Hessian matrix. The next iterate is defined as:

$$x_{t+1} = \arg\min_x Q_t(x) \quad \Rightarrow \quad x_{t+1} = x_t - [H(x_t)]^{-1}\nabla f(x_t)$$

This step jumps directly to the minimum of the local quadratic approximation. The Newton update can fail in practice primarily because the Hessian may be ill-conditioned or indefinite.

Newton's method requires strong assumptions:

- The function must be continuous C^0,
- Differentiable C^1,
- And twice differentiable C^2.

Not all continuous functions are differentiable. **All differentiable functions are continuous.** These distinctions matter for optimization guarantees.

Gradient Descent. In stochastic gradient descent (SGD), the gradient used for each update is best described as noisy but unbiased.

Convexity

A set $X \subseteq \mathbb{R}^n$ is **convex** if for any two points $x_1, x_2 \in X$, the entire line segment between them stays inside the set:

$$\forall \lambda \in [0,1], \quad \lambda x_1 + (1-\lambda)x_2 \in X.$$

A function $f : X \to \mathbb{R}$ (with **convex domain** X) is **convex** if the function value at any point on the segment between x_1 and x_2 is no larger than the linear interpolation of the endpoint values:

$$\forall x_1, x_2 \in X,\ \forall \lambda \in [0,1], \quad f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2).$$

Note: The domain X must be convex in the definition of a convex function, otherwise the interpolation point $\lambda x_1 + (1-\lambda)x_2$ may fall outside the domain.

Minimizing a **convex** function or maximizing a **concave** function guarantees that every local optimum is also a **global optimum**.

(A) 0th-order condition (Definition / Jensen form)

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2).$$

This is the core definition and does **not require derivatives**.

(B) 1st-order condition (Supporting hyperplane)

If f is **differentiable**, convexity is equivalent to:

—

$$f(x_2) \geq f(x_1) + \nabla f(x_1)^\top (x_2 - x_1), \quad \forall x_1, x_2 \in X.$$

Meaning: the tangent plane(切平面) at x_1 is a **global underestimator** of f.

(C) 2nd-order condition (Hessian test)

If f is twice differentiable, convexity is equivalent to:

$$\nabla^2 f(x) \succeq 0 \quad \text{for all } x \in X,$$

i.e., the Hessian is **positive semidefinite(PSD)** everywhere.

If a twice-differentiable function satisfies $\nabla^2 f(x) \succeq 0$ for all $x \in X$, then f is convex.

A twice-differentiable function with a positive semidefinite Hessian everywhere is convex, but the converse is false because convex functions need not be twice differentiable, and without a Hessian, the PSD condition is undefined.

Let $f(x) = x^2$ with domain $D = [-2, -1] \cup [1, 2]$. Under standard convex analysis conventions, f is convex on each interval but not convex on D as a whole.

If f and g are convex on a convex set X, then for any $\alpha, \beta \geq 0$:

$$h(x) = \alpha f(x) + \beta g(x)$$

is convex on X.

If f and g are convex on X, then

$$h(x) = \max\{f(x), g(x)\}$$

is convex on X.

Let $f \circ g = f(g(x))$.

- If g is **affine** and f is convex, then $f \circ g$ is convex ($g(x) = Ax + b$).
- More generally, if f is **convex and non-decreasing** and g is convex, then $f \circ g$ is convex.

This monotonicity condition prevents the composition from flipping curvature.

If a convex function has a local minimizer, then every local minimizer is also a global minimizer.

In general:

$$\text{Global minima} \subseteq \text{Local minima} \subseteq \text{Stationary points (for smooth } f).$$

Convexity collapses the hierarchy (for differentiable convex f):

$$\nabla f(x^*) = 0 \quad \iff \quad x^* \text{ is a global minimizer.}$$

A **saddle point** is best described as a stationary point that is neither a local minimum nor a local maximum.

Convergence of Gradient Descent

Problem Setup

We want to solve the unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} f(x)$$

using Gradient Descent (GD):

$$x_{k+1} = x_k - \alpha \nabla f(x_k),$$

where $\alpha > 0$ is the step size and x^* denotes a global minimizer. Define the **function error**

$$e_k := f(x_k) - f(x^*).$$

Core Assumptions

Convexity

A differentiable function f is convex if for all x, y:

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x).$$

This implies (for minimization) that local minima are global minima, and the tangent plane(切平面) is a global under-estimator(全局下估计).

L-Smoothness (Lipschitz Gradient)

f is **L-smooth** if its gradient is Lipschitz:

$$||\nabla f(x) - \nabla f(y)|| \leq L||x - y||.$$

A key equivalent inequality(等价不等式) (often called the **Descent Lemma(下降引理)**) is:

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

L-smoothness:

$$f(y) - f(x) = \int_0^1 \nabla f(z(t))^T (y - x)\, dt \qquad z(1) = y, z(0) = 0$$

$$f(y) - f(x) = \int_0^1 \left(\nabla f(z(t)) - \nabla f(x) + \nabla f(x)\right)^T (y - x)\, dt$$

$$f(y) - f(x) = \underbrace{\int_0^1 \nabla f(x)^T (y - x)\, dt}_{\text{Integral A}} + \underbrace{\int_0^1 \left(\nabla f(z(t)) - \nabla f(x)\right)^T (y - x)\, dt}_{\text{Integral B}}$$

$$f(y) - f(x) = \nabla f(x)^T (y - x) + \text{Integral B}$$

L-smoothness:

$$f(y) - f(x) = \int_0^1 \nabla f(z(t))^T (y - x)\, dt \qquad z(1) = y, z(0) = 0$$

$$f(y) - f(x) = \int_0^1 \left(\nabla f(z(t)) - \nabla f(x) + \nabla f(x)\right)^T (y - x)\, dt$$

$$f(y) - f(x) = \underbrace{\int_0^1 \nabla f(x)^T (y - x)\, dt}_{\text{Integral A}} + \underbrace{\int_0^1 \left(\nabla f(z(t)) - \nabla f(x)\right)^T (y - x)\, dt}_{\text{Integral B}}$$

$$f(y) - f(x) = \nabla f(x)^T (y - x) + \text{Integral B}$$

Cauchy-Schwarz inequality:

$$a \cdot b \leq |a||b|$$

$$(\nabla f(z(t)) - \nabla f(x))^T (y - x) \leq (L \cdot t\|y - x\|) \cdot \|y - x\| = Lt\|y - x\|^2$$

$$\text{Integral B} = \int_0^1 \left(\nabla f(z(t)) - \nabla f(x)\right)^T (y - x)\, dt \leq \int_0^1 Lt\|y - x\|^2\, dt$$

$$\text{Integral B} \leq \frac{L}{2}\|y - x\|^2$$

Strong Convexity

1st order condition

$$\underbrace{f(x) + \nabla f(x)^T(y-x) + \frac{\mu}{2}\|y-x\|^2}_{\text{Lower Bound (Floor)}} \leq f(y) \leq \underbrace{f(x) + \nabla f(x)^T(y-x) + \frac{L}{2}\|y-x\|^2}_{\text{Upper Bound (Ceiling)}}$$

2nd order condition $\nabla^2 f(x) \geq \mu I$, positive curvature at every x

Osculating circle [edit]

Historically, the curvature of a differentiable curve was defined through the osculating circle, which is the circle that best approximates the curve at a point. More precisely, given a point P on a curve, every other point Q of the curve defines a circle (or sometimes a line) passing through Q and tangent to the curve at P. The osculating circle is the limit, if it exists, of this circle when Q tends to P. Then the *center* and the *radius of curvature* of the curve at P are the center and the radius of the osculating circle. The curvature is the reciprocal of radius of curvature. That is, the curvature is

$$\kappa = \frac{1}{R},$$

where R is the radius of curvature[5] (the whole circle has this curvature, it can be read as turn 2π over the length $2\pi R$).

This definition is difficult to manipulate and to express in formulas. Therefore, other equivalent definitions have been introduced.

AI识图

Strong C...

Osculating circle [edit]

Historically, the curvature of a differentiable curve was defined through the osculating circle, which is the circle that best approximates the curve at a point. More precisely, given a point P on a curve, every other point Q of the curve defines a circle (or sometimes a line) passing through Q and tangent to the curve at P. The osculating circle is the limit, if it exists, of this circle when Q tends to P. Then the *center* and the *radius of curvature* of the curve at P are the center and the radius of the osculating circle. The curvature is the reciprocal of radius of curvature. That is, the curvature is

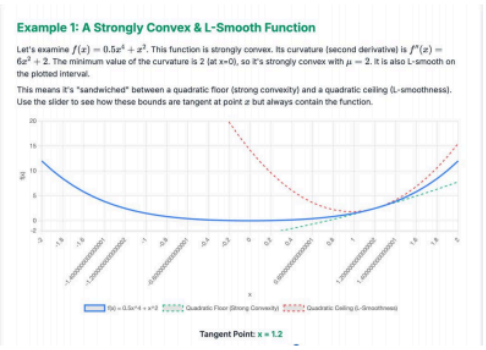
$$\kappa = \frac{1}{R},$$

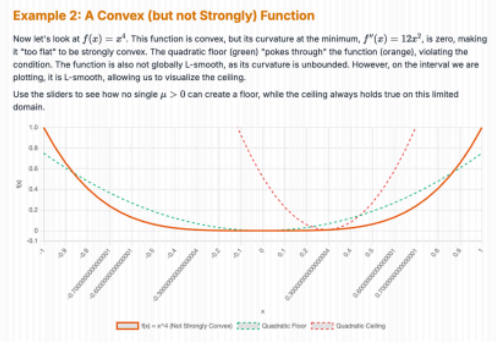
where R is the radius of curvature[5] (the whole circle has this curvature, it can be read as turn 2π over the length $2\pi R$).

This definition is difficult to manipulate and to express in formulas. Therefore, other equivalent definitions have been introduced.

$+ \frac{L}{2}\|y-x\|^2$

ing)





Guaranteed One-Step Progress (Choosing a Step Size)

Apply the Descent Lemma with $x = x_k$, $y = x_{k+1} = x_k - \alpha \nabla f(x_k)$:

$$f(x_{k+1}) \leq f(x_k) + \nabla f(x_k)^\top (x_{k+1} - x_k) + \frac{L}{2}||x_{k+1} - x_k||^2.$$

Substitute $x_{k+1} - x_k = -\alpha \nabla f(x_k)$:

$$f(x_{k+1}) \leq f(x_k) - \alpha||\nabla f(x_k)||^2 + \frac{L\alpha^2}{2}||\nabla f(x_k)||^2$$
$$= f(x_k) - \alpha\Big(1 - \frac{L\alpha}{2}\Big)||\nabla f(x_k)||^2.$$

So if $\alpha \leq \frac{1}{L}$, then $1 - \frac{L\alpha}{2} \geq \frac{1}{2}$ and we get the clean bound (taking $\alpha = \frac{1}{L}$):

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

This shows the objective is **monotonically decreasing** with that step size. Also rearrange it into:

$$||\nabla f(x_k)||^2 \leq 2L\big(f(x_k) - f(x_{k+1})\big) = 2L(e_k - e_{k+1}). \tag{1}$$

Connecting Error to Distance (The Key Bridge)

From convexity, set $x = x_k$, $y = x^*$:

$$f(x^*) \geq f(x_k) + \nabla f(x_k)^\top (x^* - x_k)$$
$$\Rightarrow e_k = f(x_k) - f(x^*) \leq \nabla f(x_k)^\top (x_k - x^*).$$

Now expand the squared distance after one GD step:

$$||x_{k+1} - x^*||^2 = ||x_k - \alpha\nabla f(x_k) - x^*||^2$$
$$= ||x_k - x^*||^2 - 2\alpha\nabla f(x_k)^\top (x_k - x^*) + \alpha^2||\nabla f(x_k)||^2.$$

Rearrange to isolate the inner product:

$$\nabla f(x_k)^\top (x_k - x^*) = \frac{1}{2\alpha}\Big(||x_k - x^*||^2 - ||x_{k+1} - x^*||^2\Big) + \frac{\alpha}{2}||\nabla f(x_k)||^2.$$

Combine with $e_k \leq \nabla f(x_k)^\top (x_k - x^*)$:

$$e_k \leq \frac{1}{2\alpha}\Big(||x_k - x^*||^2 - ||x_{k+1} - x^*||^2\Big) + \frac{\alpha}{2}||\nabla f(x_k)||^2.$$

When $\alpha = 1/L$:

$$e_k \leq \frac{L}{2}\Big(||x_k - x^*||^2 - ||x_{k+1} - x^*||^2\Big) + \frac{1}{2L}||\nabla f(x_k)||^2. \tag{2}$$

The Cancellation Trick (Telescoping)

Substitute equation (1) into equation (2):

$$e_k \leq \frac{L}{2}\Big(||x_k - x^*||^2 - ||x_{k+1} - x^*||^2\Big) + (e_k - e_{k+1}).$$

Cancel e_k from both sides:

$$e_{k+1} \leq \frac{L}{2}\Big(||x_k - x^*||^2 - ||x_{k+1} - x^*||^2\Big).$$

Sum this inequality for $k = 0$ to $K - 1$ (telescoping sum):

$$\begin{aligned}\sum_{k=0}^{K-1} e_{k+1} = \sum_{k=1}^{K} e_k &\leq \sum_{k=0}^{K-1} \frac{L}{2}\Big(||x_k - x^*||^2 - ||x_{k+1} - x^*||^2\Big) \\ &= \frac{L}{2}\Big(||x_0 - x^*||^2 - ||x_K - x^*||^2\Big) \leq \frac{L}{2}||x_0 - x^*||^2.\end{aligned}$$

Since $f(x_k)$ is decreasing, e_k is non-increasing, so $e_K \leq \frac{1}{K}\sum_{k=1}^{K} e_k$. Therefore:

$$e_K = f(x_K) - f(x*) \leq \frac{L||x_0 - x^*||^2}{2K}.$$

To ensure $e_K \leq \varepsilon$, it suffices that

$$K \geq \frac{L||x_0 - x^*||^2}{2\varepsilon},$$

which is the classical $O(1/\varepsilon)$ convergence rate for GD on convex, L-smooth functions.

Strong Convexity

If f is μ-**strongly convex**, then:

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x) + \frac{\mu}{2}||y - x||^2,$$

equivalently (when twice differentiable) $\nabla^2 f(x) \succeq \mu I$. With both L-smoothness and strong convexity, GD can achieve **linear convergence** (geometric decay) rather than $O(1/K)$.

Mini-batch SGD, Momentum, NAG, RMSProp, Adam

Mini-batch SGD (The Practical Default)

We often minimize an empirical risk

$$L(w) = \frac{1}{N}\sum_{i=1}^{N} \ell_i(w).$$

Full-batch GD uses $\nabla L(w)$, which is expensive when N is large. **Mini-batch SGD** approximates the gradient using a small batch B:

$$g_t \approx \nabla L(w_t) = \frac{1}{|B|}\sum_{i\in B} \nabla \ell_i(w_t), \qquad w_{t+1} = w_t - \alpha g_t.$$

Why Stochasticity(随机性) Can Be Preferable

In convex problems, GD has clean guarantees and the landscape is “one bowl.” In non-convex problems (typical deep nets), the landscape contains many critical points, including **saddle points**. The noise in SGD can help exploration:

- It can “kick” the iterate out of flat/unstable saddle regions.
- It can help avoid getting stuck in certain sharp basins(盆地) (informal but often observed).

Dampen the Oscillation(抑制振荡): Momentum Methods

Plain SGD can zig-zag, especially in “ravines(**深谷**)” where curvature is steep in one direction and flat in another. Momentum adds a memory of past gradients to smooth updates. One common form (velocity formulation):

$$v_{t+1} = \beta v_t + g_t, \qquad w_{t+1} = w_t - \alpha v_{t+1},$$

with $v_0 = 0$ and $\beta \in [0, 1)$ (often around 0.9–0.99).

Interpretation: v_t becomes an exponentially weighted moving average of gradients:

$$v_t = g_t + \beta g_{t-1} + \beta^2 g_{t-2} + \cdots$$

So directions that persist get amplified; directions that alternate(反复交替) (oscillations) cancel out(相互抵消).

Nesterov Accelerated Gradient (NAG): Look Ahead, Then Correct

Momentum can overshoot(超调) because it commits(笃信) strongly to the velocity. NAG modifies where you measure the gradient: evaluate it at a "lookahead" point. A typical NAG-like update:

$$\tilde{w}_t = w_t - \alpha\beta v_t, \qquad v_{t+1} = \beta v_t + \nabla L(\tilde{w}_t), \qquad w_{t+1} = w_t - \alpha v_{t+1}.$$

Intuition: "move in the momentum direction first, then compute the gradient there and correct." This often stabilizes training and can speed convergence compared to vanilla momentum.

Speeding Up Convergence: Step-size Heuristics

Move slowly in steep directions, fast in flat directions. This motivates per-parameter adaptive learning rates: scale(调整) updates differently in each coordinate(坐标分量) depending on gradient history.

RMSProp: Adaptive Step Sizes via Running Squared Gradients(平方梯度滑动量)

RMSProp keeps a moving average of squared gradients:

$$s_t = \beta s_{t-1} + (1-\beta) g_t^2 \quad (\text{elementwise square}),$$

and updates

$$w_{t+1} = w_t - \alpha \frac{g_t}{\sqrt{s_t + \varepsilon}}.$$

Effect: if a coordinate has consistently large gradients, s_t grows and the effective step size shrinks for that coordinate; if gradients are small, steps become larger.

Adam: Momentum + RMSProp (Plus Bias Correction)

Adam combines:

- a first-moment(**一阶矩**) estimate (momentum-like):

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t,$$

- a second-moment estimate (RMSProp-like):

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2.$$

Because m_t, v_t start at zero, they are biased early. Adam applies bias correction:

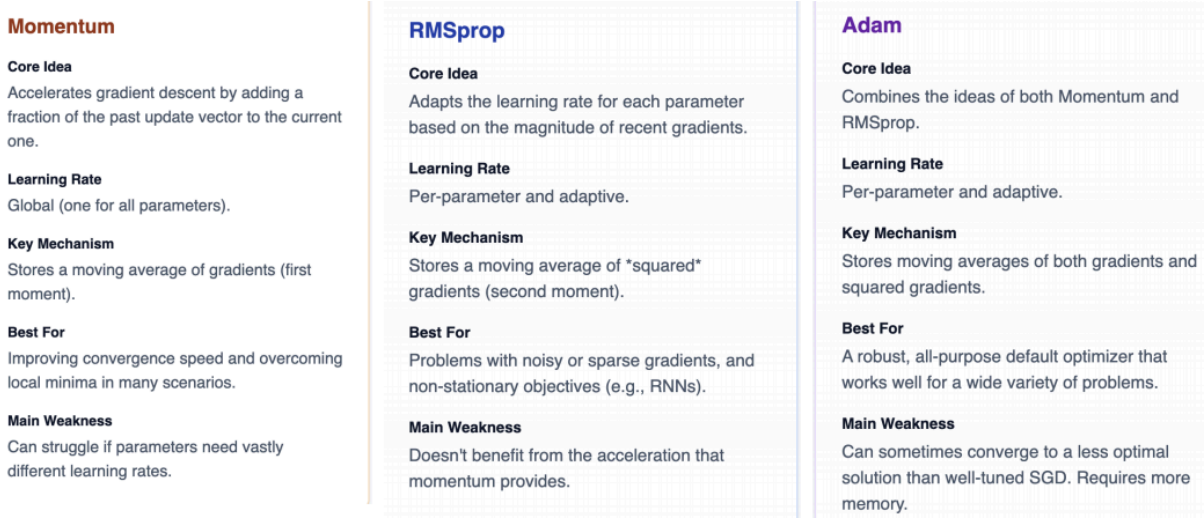
$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \qquad \hat{v}_t = \frac{v_t}{1-\beta_2^t}.$$

Update:

$$w_{t+1} = w_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}.$$

Common defaults: $\beta_1 = 0.9,\ \beta_2 = 0.999$, α tunable.

Comparison at a Glance (What problem each method targets)



Momentum

Core Idea
Accelerates gradient descent by adding a fraction of the past update vector to the current one.

Learning Rate
Global (one for all parameters).

Key Mechanism
Stores a moving average of gradients (first moment).

Best For
Improving convergence speed and overcoming local minima in many scenarios.

Main Weakness
Can struggle if parameters need vastly different learning rates.

RMSprop

Core Idea
Adapts the learning rate for each parameter based on the magnitude of recent gradients.

Learning Rate
Per-parameter and adaptive.

Key Mechanism
Stores a moving average of *squared* gradients (second moment).

Best For
Problems with noisy or sparse gradients, and non-stationary objectives (e.g., RNNs).

Main Weakness
Doesn't benefit from the acceleration that momentum provides.

Adam

Core Idea
Combines the ideas of both Momentum and RMSprop.

Learning Rate
Per-parameter and adaptive.

Key Mechanism
Stores moving averages of both gradients and squared gradients.

Best For
A robust, all-purpose default optimizer that works well for a wide variety of problems.

Main Weakness
Can sometimes converge to a less optimal solution than well-tuned SGD. Requires more memory.

Takeaway Messages

Reduced oscillation: Momentum makes the trajectory smoother, especially in “ravines(深谷)” where one direction is steep and the other is flat. It damps(抑制) the classic SGD zig-zag.

Faster convergence (often): smoothing usually creates a more direct path and builds speed in consistent downhill directions.

Nesterov’s advantage (NAG): by “looking ahead,” NAG tends to brake more appropriately near the minimum, reducing overshoot and making training more stable.

New hyperparameter: Momentum introduces β (momentum coefficient(系数)). Default values like 0.9 often work well, but the *interaction* between α and β matters—too large a learning rate plus high momentum can turn your optimizer into a pinball(弹珠) machine.

Practical Advice

Start with Adam for most deep learning tasks: it's a strong baseline because it handles scaling differences and gradient noise with minimal tuning.

Consider RMSProp in settings that are highly non-stationary or historically tricky (e.g., some RNN / RL-style training loops). (Modern practice often still uses Adam variants here, but RMSProp remains conceptually relevant.)

Don't forget SGD + Momentum: when you have time to tune and you care about best final generalization, SGD+Momentum can sometimes beat Adam—especially in supervised vision-style regimes(系统) with good schedules.

Question 3 Which of the following statements is **always** true for a real-valued data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ and $\mathbf{y} \in \mathbb{R}^n$ with $d > n$ where n is the number of data points and d represent the feature dimension?

- ☐ The sample points are linearly separable.
- ■ The data points lie in an at most n-dimensional subspace of $\mathbb{R}^d$, so there is at least one non-zero direction in $\mathbb{R}^d$ that is orthogonal to all these points.
- ☐ The weights $\mathbf{w}$ can be computed with least-squares linear regression as follows: $\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$.
- ☐ $\mathbf{X}^\top \mathbf{X}$ has exactly $d - n$ eigenvectors with eigenvalue zero.

Solution: In the case where $d > n$ (more features than samples), $\mathbf{X}^\top \mathbf{X}$ is not invertible because it is a $d \times d$ matrix with rank at most n since $\mathbf{X}$ has only n rows. When $\mathbf{X}^\top \mathbf{X}$ is not full rank, its inverse does not exist, hence the least squares solution cannot be computed with the given formula. The matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ can have at most n linearly independent rows, so its rank is at most n. As a result, the rank of the $d \times d$ matrix $\mathbf{X}^\top \mathbf{X}$ is also limited to n, implying that there can be **at most** n non-zero eigenvalues. Consequently, there are **at least** $d - n$ zero eigenvalues and the statement cannot be considered definitively true. Linear separability of the sample points is not guaranteed simply by having $d > n$, it depends on the specific data. This remaining option is true.

给定：特征数多于样本数，这是典型的 **高维小样本（overparameterized）** 场景。

❌ **选项 1**

线性可分性不是只由维度决定的。反例很简单：所有点完全重合；或者标签随机但特征退化。线性可分性取决于**数据分布和标签**，不是"维度是否大于样本数"。

✅ **选项 2**

X 的行向量最多 n 个线性无关，所以所有数据点 **最多张成一个 n 维子空间。**整个空间是 $\mathbb{R}^d$，而子空间维数 ≤ n，且 $d > n$ ➡️ **必然存在非零的正交补空间。**

存在 $v \in \mathbb{R}^d, v \neq 0$，使得：$Xv = 0$。也就是说：这个方向对所有数据点"不可见"；数据在这个方向上的投影全为 0。

❌ **选项 3**

$$X^\top X \in \mathbb{R}^{d \times d}$$

但：

$$\text{rank}(X^\top X) = \text{rank}(X) \leq n < d$$

不是满秩矩阵，不可逆，所以这个公式 **根本不成立。**

正确做法应该是：Moore–Penrose 伪逆或加正则项（Ridge）。

❌ **选项 4**

我们只知道：$\text{rank}(X^\top X) \leq n$，因此：非零特征值 **最多 n 个**零特征值 **至少 $d - n$ 个。**但注意：至少 ≠ 恰好。如果：X 的行本身就线性相关，那

么 rank(X) < n，零特征值会 **多于 (d-n)**。