1. Math Prerequisites

- Data matrix: $\boldsymbol{X} \in \mathbb{R}^{N\times D}$, weights $\boldsymbol{w} \in \mathbb{R}^{D}$, targets $\boldsymbol{y} \in \mathbb{R}^{N}$.
- Train / test errors: $\mathcal{E}_{\text{train}}$, $\mathcal{E}_{\text{test}}$ (avoid confusion with expectation $\mathbb{E}[\cdot]$).

- $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times p} \Rightarrow (\boldsymbol{AB})^T = \boldsymbol{B}^T\boldsymbol{A}^T$.
- Trace: $\text{Tr}(\boldsymbol{A}) = \sum_i A_{ii} = \sum_i \lambda_i$. $\text{Tr}(\boldsymbol{ABC}) = \text{Tr}(\boldsymbol{BCA})$.
- **Eigendecomp**: $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T$ (Symm). SVD: $\boldsymbol{A} = \boldsymbol{U}\Sigma\boldsymbol{V}^T$.
- **Gradients**: $\nabla_{\boldsymbol{x}}(\boldsymbol{b}^T\boldsymbol{x}) = \boldsymbol{b}$, $\nabla_{\boldsymbol{x}}\|\boldsymbol{x}\|_2^2 = 2\boldsymbol{x}$.
- **Quadratic Form**: $f(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$.
  - $\nabla_{\boldsymbol{x}} f = (\boldsymbol{A} + \boldsymbol{A}^T)\boldsymbol{x}$. (If symm: $2\boldsymbol{A}\boldsymbol{x}$), $\nabla^2 f = \boldsymbol{A} + \boldsymbol{A}^T$.
- **Hessian $\boldsymbol{H}$**: Matrix of 2nd derivatives, $H_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$.
  - **Taylor (2nd order)**: $f(\boldsymbol{x} + \boldsymbol{d}) \approx f(\boldsymbol{x}) + \nabla f^T\boldsymbol{d} + \frac{1}{2}\boldsymbol{d}^T\boldsymbol{H}\boldsymbol{d}$.
  - **Curvature**: Eigenvalues of $\boldsymbol{H}$ determine shape.
  - $\boldsymbol{H} \succ 0$ (PD) $\Rightarrow$ Local Min (Bowl), $\boldsymbol{H} \prec 0$ (ND) $\Rightarrow$ Local Max (Hill).
  - Indefinite (mixed signs) $\Rightarrow$ Saddle Point.

- Bayes: $P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} = \frac{P(X|Y)P(Y)}{\sum_y P(X|y)P(y)}$. If $X \perp\!\!\!\perp Y$,
- **Expectation**: $\mathbb{E}[aX + b] = a\mathbb{E}[X] + b$. then $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$.
- **Variance**: $\text{Var}(X) = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$, $\text{Var}(aX + b) = a^2\,\text{Var}(X)$.
- **Gaussian**: $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \Sigma) = \frac{\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)}{(2\pi)^{D/2}|\Sigma|^{1/2}}$.

Linear Algebra Checks (Exam '24)
- **Invertibility**: For $\boldsymbol{X} \in \mathbb{R}^{N\times D}$, $\boldsymbol{X}^T\boldsymbol{X}$ is $D \times D$.
- If $D > N$ (High dim), $\text{rank}(\boldsymbol{X}) \le N < D$. $\boldsymbol{X}^T\boldsymbol{X}$ is singular (not invertible). LS solution not unique.
- **Variance Transformation**: $\text{Var}(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}) = \boldsymbol{A}\,\text{Var}(\boldsymbol{x})\boldsymbol{A}^T$. (1D: $a^2\sigma^2$).

2. Statistical Learning Theory

Bias-Variance Decomposition (MSE) True model: $y = f(x) + \epsilon, \mathbb{E}[\epsilon] = 0, \text{Var}(\epsilon) = \sigma^2$. Estimator $\hat{f}_D(x)$ trained on dataset D. Expectation is over dataset D (and noise), with fixed x: $\mathbb{E}_D\left[(y - \hat{f}_D(x))^2\right] = \text{Bias}^2(\hat{f}) + \text{Var}(\hat{f}) + \sigma^2$, $\text{Bias}^2(\hat{f}) = \left(\mathbb{E}_D[\hat{f}_D(x)] - f(x)\right)^2$, $\text{Var}(\hat{f}) = \mathbb{E}_D\left[\left(\hat{f}_D(x) - \mathbb{E}_D[\hat{f}_D(x)]\right)^2\right]$. **Overfitting (high Var)**: $\mathcal{E}_{\text{train}} \ll \mathcal{E}_{\text{test}}$. **Underfitting (high Bias)**: both errors high. Regularization: Var $\downarrow$, Bias $\uparrow$. More data: Var $\downarrow$.

Exact Formula Check (Exam) $\mathbb{E}[(Y - \hat{f})^2] = \text{Bias}[\hat{f}]^2 + \text{Var}[\hat{f}] + \sigma^2$. *Note:* Irreducible error σ^2 cannot be removed.

Train / Test / Validation
- **Standardization**: $\tilde{x}_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}$.
- **Important**: compute μ, σ using train only; apply to val/test.

Bayes Risk & Calibration (Theory) **Bayes Opt. Classifier**: $f^*(x) = \text{sign}(2\eta(x) - 1)$ where $\eta(x) = P(Y = 1|X = x)$. **Bayes Risk**: $\mathcal{L}^* = \mathbb{E}[\min(\eta(X), 1 - \eta(X))]$. **Calibration**: Minimizing surrogate ϕ-risk leads to f^* if ϕ convex, diff. at 0, $\phi'(0) < 0$.
- **Square**: $g^*(x) = 2\eta(x) - 1$. (Calibrated 已校准) **Logistic**: $g^*(x) = \log\frac{\eta(x)}{1-\eta(x)}$. (Calibrated) **Hinge**: $g^*(x) = \text{sign}(2\eta(x) - 1)$. (Calibrated)

3. Likelihood & MLE Principles

Definitions & Properties Given i.i.d. data $D = \{x_1, \ldots, x_N\}$ and model param θ.
- **Likelihood**: $L(\theta) = P(D|\theta) = \prod_{n=1}^N P(x_n|\theta)$. (Func of θ, not x).
- **Log-Likelihood (LL)**: $\ell(\theta) = \ln L(\theta) = \sum_{n=1}^N \ln P(x_n|\theta)$.
- **MLE**: $\hat{\theta}_{MLE} = \arg\max_\theta L(\theta) \equiv \arg\max_\theta \ell(\theta) \equiv \arg\min_\theta(-\ell(\theta))$.
- **Equivalence**: Constants (e.g., $\frac{1}{b}$) or scaling do NOT change $\arg\max$. (Exam '23 Q3).

Properties of MLE (Handwrite)
- **Consistent**: Converges to true θ^* as $N \to \infty$.
- **Asymptotic Normality**: $\hat{\theta} \approx \mathcal{N}(\theta^*, \mathcal{I}(\theta^*)^{-1})$.
- **Efficiency**: Achieves Cramér-Rao lower bound (lowest variance).
- $\mathcal{I}(\theta)$ is Fisher Info.

Key Connections (Derivations)
- **Gaussian $\to$ MSE**: If noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$, then $y_n \sim \mathcal{N}(\boldsymbol{w}^T\boldsymbol{x}_n, \sigma^2)$.

$$\max \sum \ln\left(C \cdot e^{-\frac{(y-\hat{y})^2}{2\sigma^2}}\right) \iff \min \sum (y_n - \boldsymbol{w}^T\boldsymbol{x}_n)^2$$

Maximizing Gaussian LL is equivalent to Minimizing MSE (Least Squares).

- **Bernoulli $\to$ Logistic Loss**: If $y \in \{0, 1\}$, $P(y|\boldsymbol{x}) = \hat{y}^y(1 - \hat{y})^{1-y}$.

$$-\ln L(\boldsymbol{w}) = -\sum [y_n \ln \hat{y}_n + (1 - y_n)\ln(1 - \hat{y}_n)]$$

Maximizing Bernoulli LL is equivalent to Minimizing Cross-Entropy.

4. Linear Regression

Data $D = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$, $\boldsymbol{x}_n \in \mathbb{R}^D$ (optionally augmented with 1). Model: $y_n = \boldsymbol{w}^T\boldsymbol{x}_n + \epsilon_n$.
Least Squares (MSE) $L(\boldsymbol{w}) = \frac{1}{2N}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$ **Gradient**: $\nabla_{\boldsymbol{w}} L = \frac{1}{N}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$ **Normal Equation**: $\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{y}$.
If $\boldsymbol{X}^T\boldsymbol{X}$ invertible (full column rank): $\boldsymbol{w}^* = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$. **Invariance**: Centering/Scaling features does NOT change prediction performance for unregularized OLS.

Ridge Regression (L2) $\hat{\boldsymbol{w}}_{\text{Ridge}} = \arg\min_{\boldsymbol{w}} \frac{1}{2N}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$
$\boldsymbol{w}^* = (\boldsymbol{X}^T\boldsymbol{X} + N\lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}$.
- Always unique since $\boldsymbol{X}^T\boldsymbol{X} + N\lambda\boldsymbol{I} \succ 0$ for $\lambda > 0$.
- MAP view: Gaussian prior $\boldsymbol{w} \sim \mathcal{N}(0, \tau^2\boldsymbol{I})$ gives $\lambda \propto 1/\tau^2$ (up to scaling conventions).

Lasso Regression (L1) $\hat{\boldsymbol{w}}_{\text{Lasso}} = \arg\min_{\boldsymbol{w}} \frac{1}{2N}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{w}\|_1$
- **Sparsity**: promotes $w_j = 0$ (feature selection).
- **Subgradient**: $\partial|w_j| = \{-1\}$ if $w_j < 0$, $\{1\}$ if $w_j > 0$, $[-1, 1]$ if $w_j = 0$.

Perceptron Convergence Proof Assumptions: Data $\|\boldsymbol{x}_n\| \le R$, Linear Sep: $y_n\boldsymbol{w}_*^T\boldsymbol{x}_n \ge \gamma$. **Algorithm**: If err, $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + y_i\boldsymbol{x}_i$. (Init $\boldsymbol{w}_0 = \boldsymbol{0}$).
- **Lower Bound (Dot Prod)**: $\boldsymbol{w}_*^T\boldsymbol{w}_{t+1} = \boldsymbol{w}_*^T\boldsymbol{w}_t + \underbrace{y_i\boldsymbol{w}_*^T\boldsymbol{x}_i}_{\ge\gamma} \ge \boldsymbol{w}_*^T\boldsymbol{w}_t + \gamma \implies \boldsymbol{w}_*^T\boldsymbol{w}_t \ge t\gamma$.
- **Upper Bound (Norm)**: $\|\boldsymbol{w}_{t+1}\|^2 = \|\boldsymbol{w}_t\|^2 + \underbrace{2y_i\boldsymbol{w}_t^T\boldsymbol{x}_i}_{\le 0\text{(mistake)}} + \underbrace{\|y_i\boldsymbol{x}_i\|^2}_{\le R^2} \le \|\boldsymbol{w}_t\|^2 + R^2 \implies \|\boldsymbol{w}_t\|^2 \le tR^2$.
- **Conclusion (Steps t)**: $t^2\gamma^2 \le (\boldsymbol{w}_*^T\boldsymbol{w}_t)^2 \le \|\boldsymbol{w}_*\|^2\|\boldsymbol{w}_t\|^2 \le \|\boldsymbol{w}_*\|^2 tR^2 \implies t \le \frac{R^2\|\boldsymbol{w}_*\|^2}{\gamma^2}$. (Novikoff's Thm).

5. Optimization (优化算法)

Gradient Descent (GD, 梯度下降) 更新公式: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \gamma\nabla L(\boldsymbol{w}_t)$.
- If L convex (凸) and γ small enough, converges to global optimum.
- **Cost per step (全量计算代价)**: $O(ND)$. (N samples, D dims).

Projected Gradient Descent For constrained opt (带约束优化) $\min_{w\in C} L(w)$: $\boldsymbol{w}_{t+1} = \Pi_C(\boldsymbol{w}_t - \gamma\nabla L(\boldsymbol{w}_t))$
- Π_C: **Projection Operator (投影算子)**. Maps point back to valid set C (e.g., if $\|\boldsymbol{w}\| > 1$, clip it to boundary).

Stochastic Gradient Descent (SGD, 随机梯度下降)
- Sample n (随机抽样); $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \gamma\nabla L_n(\boldsymbol{w}_t)$.
- **Cost per step**: $O(D)$ (Fast!); noisy gradients (震荡大).
- Typically use **decaying LR (学习率衰减)** γ_t (or schedules).

Optimization Variants (变种)
- **Momentum (动量法)**: Accumulate velocity (累积速度/惯性).
  $\boldsymbol{m}_{t+1} = \beta\boldsymbol{m}_t + (1 - \beta)\boldsymbol{g}_t$; $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \gamma\boldsymbol{m}_{t+1}$
- **Adam**: Adaptive moments (自适应矩估计).
  $\boldsymbol{m}_{t+1} = \beta_1\boldsymbol{m}_t + (1 - \beta_1)\boldsymbol{g}_t$ (1st moment/Mean 均值)
  $\boldsymbol{v}_{t+1} = \beta_2\boldsymbol{v}_t + (1 - \beta_2)\boldsymbol{g}_t^2$ (2nd moment/Var 方差)
  $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \gamma\hat{\boldsymbol{m}}_{t+1}/(\sqrt{\hat{\boldsymbol{v}}_{t+1}} + \epsilon)$ (Scale by std dev).
- **Sign-SGD**: Direction only (仅用符号). $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \gamma\text{sign}(\boldsymbol{g}_t)$.

Subgradient Definition (次梯度 - 解决不可导) For convex non-diff function f (e.g., L1 norm $|x|$), $\boldsymbol{g}$ is a subgradient at $\boldsymbol{w}$ if: $f(\boldsymbol{u}) \ge f(\boldsymbol{w}) + \boldsymbol{g}^T(\boldsymbol{u} - \boldsymbol{w})$, $\forall\boldsymbol{u}$ *Meaning*: The tangent plane is always **below** the function (切线在函数下方).
Newton's Method (牛顿法) $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \gamma[\boldsymbol{H}(\boldsymbol{w}_t)]^{-1}\nabla L(\boldsymbol{w}_t)$
- **Quadratic convergence (二次收敛)**: Very fast near optimum.
- **Cost**: Hessian inversion is $O(D^3)$. (Too slow for large dims).

6. Logistic Regression

Model: $\hat{y} = \sigma(z)$, $z = \boldsymbol{w}^T\boldsymbol{x}$. **Sigmoid Deriv**: $\sigma' = \sigma(1 - \sigma)$.
Loss (NLL): $L(\boldsymbol{w}) = -\sum[y_n\log\hat{y}_n + (1 - y_n)\log(1 - \hat{y}_n)]$.
Gradient: $\nabla_{\boldsymbol{w}} L = \frac{1}{N}\sum(\hat{y}_n - y_n)\boldsymbol{x}_n = \frac{1}{N}\boldsymbol{X}^T(\hat{\boldsymbol{y}} - \boldsymbol{y})$.
Hessian Derivation: Chain rule on error term $(\hat{y}_n - y_n)$:
$\frac{\partial}{\partial\boldsymbol{w}}(\ldots) = \frac{\partial\hat{y}_n}{\partial z_n}\boldsymbol{x}_n = \hat{y}_n(1 - \hat{y}_n)\boldsymbol{x}_n = s_n\boldsymbol{x}_n$ (Scalar s_n · Vector $\boldsymbol{x}_n$)
Matrix Form: Summing $s_n\boldsymbol{x}_n\boldsymbol{x}_n^T$ over N samples:
$\boldsymbol{H} = \frac{1}{N}\boldsymbol{X}^T\boldsymbol{S}\boldsymbol{X}$ where $\boldsymbol{S} = \text{diag}(s_1, \ldots, s_N)$.

- **Convexity**: $\forall\boldsymbol{v} \ne \boldsymbol{0}, \boldsymbol{v}^T\boldsymbol{H}\boldsymbol{v} = \frac{1}{N}\sum s_n(\boldsymbol{v}^T\boldsymbol{x}_n)^2 \ge 0 \implies$ Global Min.
- **Newton Update**: $\boldsymbol{w} \leftarrow \boldsymbol{w} - (\boldsymbol{X}^T\boldsymbol{S}\boldsymbol{X})^{-1}\boldsymbol{X}^T(\hat{\boldsymbol{y}} - \boldsymbol{y})$.

Formal Convexity Proof Steps (Exam '24) To prove convexity, show Hessian $\boldsymbol{H} \succeq 0$.
- 1. Derive Grad: $\nabla L = \frac{1}{N}\boldsymbol{X}^T(\hat{\boldsymbol{y}} - \boldsymbol{y})$.
- 2. Derive Hessian: $\boldsymbol{H} = \frac{1}{N}\boldsymbol{X}^T\boldsymbol{S}\boldsymbol{X}$ with $\boldsymbol{S} = \text{diag}(\hat{y}_n(1 - \hat{y}_n))$.
- 3. Proof: For any vector $\boldsymbol{v}$, $\boldsymbol{v}^T\boldsymbol{H}\boldsymbol{v} = \frac{1}{N}\boldsymbol{v}^T\boldsymbol{X}^T\boldsymbol{S}\boldsymbol{X}\boldsymbol{v} = \frac{1}{N}(\boldsymbol{X}\boldsymbol{v})^T\boldsymbol{S}(\boldsymbol{X}\boldsymbol{v})$.
- 4. Conc: Since $\hat{y}(1 - \hat{y}) > 0$, $\boldsymbol{S}$ is positive definite diagonal, so quadratic form ≥ 0.

7. SVM & Kernels

Soft-Margin SVM (Primal) obj: Minimize *Hinge Loss* + L_2 Regularization:
$J(\boldsymbol{w}, b) = \frac{1}{N}\sum_{n=1}^N \underbrace{\max(0, 1 - y_n(\boldsymbol{w}^T\boldsymbol{x}_n + b))}_{\text{Hinge Loss}} + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$

SGD Updates (Pick random $(\boldsymbol{x}_n, y_n)$):
- Case 1 ($y_nf(\boldsymbol{x}_n) < 1$): Violation or inside margin. $\boldsymbol{w} \leftarrow (1 - \gamma\lambda)\boldsymbol{w} + \gamma y_n\boldsymbol{x}_n$.
- Case 2 ($y_nf(\boldsymbol{x}_n) \ge 1$): Correct & safe. $\boldsymbol{w} \leftarrow (1 - \gamma\lambda)\boldsymbol{w}$ (Weight decay only)

Note: $C \propto \frac{1}{\lambda}$. Small λ / Large $C \to$ Hard Margin (Overfit risk).
Dual Formulation (Kernelized) $\max_\alpha \sum_{n=1}^N \alpha_n - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ **Constraints**: $0 \le \alpha_n \le C$ AND $\sum\alpha_n y_n = 0$.
KKT Conditions & Support Vectors (SVs)
- $\alpha_n = 0$: Correct ($y_nf(\boldsymbol{x}_n) > 1$). Not an SV.
- $\alpha_n > 0$: **Support Vector**.
  - $0 < \alpha_n < C$: **On Margin** ($y_nf(\boldsymbol{x}_n) = 1$). **Use for** b^*!
  - $\alpha_n = C$: **Inside Margin/Error** ($y_nf(\boldsymbol{x}_n) < 1$).
- $\boldsymbol{w}^* = \sum\alpha_n y_n\boldsymbol{x}_n$ (Exists only if linear kernel).
- **Bias b^***: Average over SVs with $0 < \alpha_k < C$: $b^* = \frac{1}{|S|}\sum_{k\in S}(y_k - \sum_{m\in SV}\alpha_m y_m k(\boldsymbol{x}_m, \boldsymbol{x}_k))$
- **Predict**: $y_{\text{new}} = \text{sign}\left(\sum_{n\in SV}\alpha_n y_n k(\boldsymbol{x}_n, \boldsymbol{x}_{\text{new}}) + b^*\right)$.

Representer Theorem (Handwrite) For any loss L and regularizer $\Omega(\|w\|_2)$ (strictly monotonic), the optimal solution $\boldsymbol{w}^*$ lies in the span of the data:

$$\boldsymbol{w}^* = \sum_{n=1}^N \alpha_n\boldsymbol{x}_n = \boldsymbol{X}^T\boldsymbol{\alpha}$$

This allows kernelizing linear models: $\boldsymbol{w}^T\boldsymbol{x} = \sum\alpha_n\langle\boldsymbol{x}_n, \boldsymbol{x}\rangle = \sum\alpha_n k(\boldsymbol{x}_n, \boldsymbol{x})$.
Kernels & Mercer's Theorem Def: $k(\boldsymbol{x}, \boldsymbol{x}') = \langle\phi(\boldsymbol{x}), \phi(\boldsymbol{x}')\rangle$. Valid iff Gram Matrix $\boldsymbol{K} \succeq 0$.
Common Kernels:
- **Linear**: $\boldsymbol{x}^T\boldsymbol{x}'$. **Poly**: $(\gamma\boldsymbol{x}^T\boldsymbol{x}' + c)^d$.
- **RBF (Gaussian)**: $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\gamma\|\boldsymbol{x} - \boldsymbol{x}'\|^2)$.
  - $\gamma \approx \frac{1}{2\sigma^2}$. **Large** γ: Narrow peak $\to$ Overfitting.
  - **Small** γ: Flat $\to$ Underfitting (Linear-like).

Kernel Construction Rules (Closure): If k_1, k_2 are valid kernels, $c > 0, f(\cdot)$ polynomial with positive coefficients:
- Sum: $k_1 + k_2$ Product: $k_1 \cdot k_2$ Scale: $c \cdot k_1$
- Mapping: $k(\phi(\boldsymbol{x}), \phi(\boldsymbol{x}'))$ Exp: $\exp(k_1)$
- Poly: $f(k_1)$ (e.g., $k_1^2 + 3k_1 + 1$)

Disproving Validity (Exam '24) To show $k(\boldsymbol{x}, \boldsymbol{y})$ is **invalid**, find set $\{\boldsymbol{x}_1, \boldsymbol{x}_2\}$ s.t. Gram Matrix $\boldsymbol{K}$ is not PSD ($\det(\boldsymbol{K}) < 0$). **Example**: $k(x, y) = (xy + c)^d$ with $c < 0$. Pick $x_1 = 1, x_2 = -1$.
$\boldsymbol{K} = \begin{pmatrix}(1 + c)^d & (c - 1)^d \\ (c - 1)^d & (1 + c)^d\end{pmatrix}$. Check if $\det(\boldsymbol{K}) = (1 + c)^{2d} - (c - 1)^{2d} < 0$.

8. Unsupervised Learning

PCA Centered data. Cov $\boldsymbol{S} = \frac{1}{N}\boldsymbol{X}^T\boldsymbol{X}$. $\max_{\boldsymbol{u}} \boldsymbol{u}^T\boldsymbol{S}\boldsymbol{u}$ s.t. $\boldsymbol{u}^T\boldsymbol{u} = 1$ Principal components: eigenvectors of $\boldsymbol{S}$ with largest eigenvalues.
K-Means Objective: $J = \sum_{k=1}^K\sum_{n\in C_k}\|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|_2^2$
1. Initialize $\boldsymbol{\mu}_k$. 2. Assign $z_n = \arg\min_k\|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|_2^2$. 3. Update $\boldsymbol{\mu}_k = \frac{1}{|C_k|}\sum_{n\in C_k}\boldsymbol{x}_n$.

GMM + EM: $p(\boldsymbol{x}) = \sum_{k=1}^K \pi_k\,\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \Sigma_k)$
- E-step: $\gamma_{nk} = p(z_n = k|\boldsymbol{x}_n) = \frac{\pi_k\mathcal{N}(\boldsymbol{x}_n|\theta_k)}{\sum_j\pi_j\mathcal{N}(\boldsymbol{x}_n|\theta_j)}$.
- M-step: update $\pi_k, \boldsymbol{\mu}_k, \Sigma_k$ via weighted MLE.

Weighted GMM Updates Obj: $\max\sum w_n\log\sum\pi_k\mathcal{N}$. E-Step q_{nk} same as std. **M-Step**: $\mu_k = \frac{\sum w_n q_{nk} x_n}{\sum w_n q_{nk}}$, $\Sigma_k = \frac{\sum w_n q_{nk}(x_n - \mu_k)(x_n - \mu_k)^T}{\sum w_n q_{nk}}$, $\pi_k = \frac{\sum w_n q_{nk}}{\sum w_n}$.
GMM Free Parameters Count (Exam '22) Total params for K clusters in D dim:

- **Full Covariance**: $K-1$ (weights) $+KD$ (means) $+K\frac{D(D+1)}{2}$ (cov).
- **Spherical** ($\sigma^2 I$): $K-1+KD+K=KD+2K-1$.

9. Neural Networks

1. Backpropagation (The δ Rule) Forward: $\boldsymbol{z}^{(l)}=\boldsymbol{W}^{(l)}\boldsymbol{a}^{(l-1)}+\boldsymbol{b}^{(l)}$, $\boldsymbol{a}^{(l)}=\phi(\boldsymbol{z}^{(l)})$.
Backward: Propagate error $\boldsymbol{\delta}^{(l)}=\frac{\partial L}{\partial \boldsymbol{z}^{(l)}}$.

- Output: $\boldsymbol{\delta}^L=\nabla_{\boldsymbol{a}}L\odot\phi'(\boldsymbol{z}^L)$.
- Hidden: $\boldsymbol{\delta}^l=(\boldsymbol{W}^{l+1})^T\boldsymbol{\delta}^{l+1}\odot\phi'(\boldsymbol{z}^l)$.
- Grads: $\frac{\partial L}{\partial \boldsymbol{W}^l}=\boldsymbol{\delta}^l(\boldsymbol{a}^{l-1})^T$, $\frac{\partial L}{\partial \boldsymbol{b}^l}=\boldsymbol{\delta}^l$.

Matrix Shapes (Exam '22) Layer $\boldsymbol{Y}=\boldsymbol{X}\boldsymbol{W}+\boldsymbol{b}$. Given $\boldsymbol{\delta}_Y=\partial L/\partial \boldsymbol{Y}$.

- $\frac{\partial L}{\partial \boldsymbol{W}}=\boldsymbol{X}^T\boldsymbol{\delta}_Y$ $\quad\frac{\partial L}{\partial \boldsymbol{X}}=\boldsymbol{\delta}_Y\boldsymbol{W}^T$ $\quad\frac{\partial L}{\partial \boldsymbol{b}}=\text{sum}_{rows}(\boldsymbol{\delta}_Y)$.

Universal Approximation (Barron) A single hidden layer NN with sufficiently many neurons and non-polynomial activation can approximate any continuous function on a compact set.

2. Complexity: **Width** H vs. **Depth** L, For Fully Connected layers:

- **Widening** ($H\to 2H$): Params/Ops $\propto H^2$. (Quad. cost).
- **Deepening** ($L\to 2L$): Params/Ops $\propto L$. (Linear cost).
- **Result**: Deep & Narrow is computationally cheaper than Shallow & Wide for same param count.

3. Softmax Properties Given logits $\boldsymbol{x}$, prediction is $\arg\max_k x_k$.

- **Shift** ($\boldsymbol{x}\to\boldsymbol{x}+b$): Probs unchanged. **Acc & Loss Unchanged.**
- **Scale** ($\boldsymbol{x}\to\alpha\boldsymbol{x},\alpha>0$): Order preserved.
  - **Accuracy**: Unchanged.
  - **Loss**: Changes! $\alpha>1$ (sharp) $\to$ Loss $\downarrow$. $\alpha<1$ (flat) $\to$ Loss $\uparrow$.

4. Regularization & Init

- **Xavier**: $\text{Var}(W)\approx\frac{2}{n_{in}+n_{out}}$ (Sigmoid). **He**: $\frac{2}{n_{in}}$ (ReLU).
- **Batch Norm**: $\frac{x-\mu_B}{\sigma_B}\cdot\gamma+\beta$. (Dep. on batch).
- **Layer Norm**: $\frac{x_{nk}-\mu_n}{\sigma_n}$. Indep of batch. (For Transf.)
- **Dropout**: Train: mask w/ p. Test: scale weights by $(1-p)$.

Activation Derivatives GeLU: $\phi(z)=z\sigma(cz)=z\frac{1}{1+e^{-cz}}$. ($c\approx1.702$).

- **Deriv**: $\phi'(z)=\sigma(cz)+z\cdot\sigma(cz)(1-\sigma(cz))\cdot c$.
- **Limits**: $z\to\infty$: $\phi'(z)\to1+0=1$ (Like ReLU). $z\to-\infty$: $\phi'(z)\to0$ (Like ReLU).
- **Relation**: Smooth approx of ReLU. Non-monotonic.

10. Convolutional Neural Networks (CNNs)

Dimensions & Layers (尺寸计算) Output Size (输出边长): $W_{out}=\lfloor\frac{W_{in}-K+2P}{S}\rfloor+1$
Output Depth (输出深度): C_{out} = number of filters (滤波器数量)

- W_{in}: Input Size (输入边长); K: Kernel Size (卷积核大小)
- P: Padding (填充); S: Stride (步长)
- **Stride** (S): Step size (滑窗步长).
- **Pooling (池化)**: W,H reduced by S (尺寸减小); C unchanged (通道不变).
- **Flatten (展平)**: 3D vol $\to$ 1D vec. Size: $(W\cdot H\cdot C)$.

Learnable Parameters (参数量 - Exam '24)

- **Conv Layer**: $\underbrace{(K^2\cdot C_{in}+1)}_{\text{weights+bias per filter}}\cdot\underbrace{C_{out}}_{\text{\# filters}}$.
- **FC Layer**: $(N_{in}+1)\cdot N_{out}$. (+1 for bias 偏置).
- **Pool Layer**: 0 parameters (无参数).

Architecture Insights

- **ResNet**: $y=F(x)+x$; identity shortcut (恒等跳跃) eases opt.
- **Receptive Field (感受野)**: Input area "seen" by pixel. $\uparrow$ with depth.
- **1x1 Conv**: Changes channel C (调整通道数); keeps W,H.

11. Transformers & NLP

1. Self-Attention $\text{Attention}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V})=\text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)\boldsymbol{V}$

- $\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}$ from same source (Self-Attn). $Q\in\mathbb{R}^{S\times d_k}$. 这里是 d 下标 k。
- **Scale** $\sqrt{d_k}$: Prevents dot product from blowing up $\to$ prevents softmax saturation $\to$ prevents vanishing gradients. **Complexity**: $O(S^2d+Sd^2)$. S^2 **is bottleneck** (Attn Matrix). vs Linear Attn $O(S)$.

2. Positional Embeddings (PE, 位置编码)

- Transformer w/o PE is **Permutation Equivariant (置换等变性)** $\to$ acts like a **Set Function**.
- **考点**: w/o PE, it treats input as a "bag of words".
  - If inputs are identical ($x_i=x_j$), outputs are identical ($z_i=z_j$), regardless of position.
  - Cannot solve order-tasks (e.g., "output 0 at odd pos" 奇数位输出 0).
- PE is **added (按元素相加)** (not concat) to input embeddings.
- $E_{final}=E_{word}+E_{pos}$. (Dimension stays d_{model}).

3. BERT vs. GPT (Architecture)

- **BERT (Encoder)**: Masked LM. Predicts masks **in parallel**.
  - **Independence Assumption**: Predicts $P(w_A,w_B|C)\approx P(w_A|C)P(w_B|C)$. Ignores dependency between masked tokens.
- **GPT (Decoder)**: Autoregressive. Sequential generation.
  - **Masked Attn**: Enforces causality (cant see future).

4. **Theoretical Limits** Transformer has bounded computation per token ($O(1)$ depth). Cannot solve problems requiring linear time $\Omega(i)$ w.r.t input index.

12. Generative AI

VAE (Variational Autoencoder) Objective: Maximize ELBO (Evidence Lower Bound). $\mathcal{L}=\underbrace{\mathbb{E}_q[\log p(x|z)]}_{\text{Reconstruction}}-\underbrace{D_{\text{KL}}(q(z|x)\parallel p(z))}_{\text{Regularization}}$

- **Rec. Term**: Ensure output $\approx$ input.
- **KL Term**: Force latent z to be $\mathcal{N}(0,I)$.
- **Reparameterization Trick**: $z=\mu+\sigma\odot\epsilon$ (allows backprop).

Diffusion Models (DDPM)

- **Forward (No train)**: $q(x_t|x_{t-1})=\mathcal{N}(x_t;\sqrt{1-\beta_t}x_{t-1},\beta_tI)$. Property: $q(x_t|x_0)=\mathcal{N}(x_t;\sqrt{\bar\alpha_t}x_0,(1-\bar\alpha_t)I)$.
- **Reverse (Train)**: Train NN ϵ_θ to **predict the noise**. Approx $q(x_{t-1}|x_t)$ with $p_\theta(x_{t-1}|x_t)$. $\mu_\theta(x_t,t)=\frac{1}{\sqrt{\alpha_t}}\left(x_t-\frac{1-\alpha_t}{\sqrt{1-\bar\alpha_t}}\epsilon_\theta(x_t,t)\right)$.
- **Loss (MSE)**: $\mathcal{L}=\|\epsilon-\epsilon_\theta(x_t,t)\|^2$. Score Matching $\approx\nabla\log p(x_t)$.

Score Matching Derivation True Score $\nabla_{\boldsymbol{x}_t}\log p(\boldsymbol{x}_t)$ is intractable. **Tractable form**: Condition on $\boldsymbol{x}_0$. $\nabla_{\boldsymbol{x}_t}\log p(\boldsymbol{x}_t|\boldsymbol{x}_0)=\nabla\log\mathcal{N}(\boldsymbol{x}_t;\sqrt{\bar\alpha_t}\boldsymbol{x}_0,(1-\bar\alpha_t)\boldsymbol{I})=-\frac{\boldsymbol{x}_t-\sqrt{\bar\alpha_t}\boldsymbol{x}_0}{1-\bar\alpha_t}=-\frac{\sqrt{1-\bar\alpha_t}\boldsymbol{\epsilon}}{\sqrt{1-\bar\alpha_t}^2}=-\frac{\boldsymbol{\epsilon}}{\sqrt{1-\bar\alpha_t}}$. $\implies$ Matching score is equiv to predicting noise $\boldsymbol{\epsilon}$.

GAN (Generative Adversarial Nets) Min-Max Game (Zero-sum): $\uparrow$ see up

- **D (Discriminator)**: Maximize. Real $x\to1$, Fake $G(z)\to0$.
- **G (Generator)**: Minimize. Trick D so $D(G(z))\to1$.
- **Issue**: Mode collapse, unstable training.

13. Metrics & Misc

Metrics: $P=TP/(TP+FP)$, $R=TP/(TP+FN)$, $F1=2PR/(P+R)$
Conf Mat: $y\to\hat y$: $1\to1$ (TP), $0\to0$ (TN), $0\to1$ (FP/Type I), $1\to0$ (FN/Type II). **混淆矩阵**: 预测 $\hat y=1$: 对是 TP/错是 FP(Type I); 预测 $\hat y=0$: 错是 FN(Type II)/对是 TN.

Fairness Criteria Variables: A (sensitive), Y (true), $\hat Y$ (pred). (xxx same).

- **1. Independence**: $\hat Y\perp\!\!\!\perp A$. $P(\hat Y=1|A=a)=P(\hat Y=1|A=b)$. (Rate $\hat Y$).
- **2. Separation**: $\hat Y\perp\!\!\!\perp A\mid Y$. $P(\hat Y=1|Y=y,A=a)=P(\hat Y=1|Y=y,A=b)$. (TPR/FPR).
- **3. Sufficiency**: $Y\perp\!\!\!\perp A\mid\hat Y$. $P(Y=1|\hat Y=s,A=a)=P(Y=1|\hat Y=s,A=b)$. (PPV).

Impossibility Thm: If base rates differ ($P(Y|A)\neq P(Y|A')$) and predictor imperfect, cannot satisfy all 3 simultaneously.

Mitigation Strategies (Handwrite)

- **Pre-processing**: Adjust features x to be uncorrelated with A.
- **In-processing**: Add regularization term to loss during training.
- **Post-processing**: Adjust thresholds/outputs of learned classifier.

14. Adversarial Robustness

Goal: **Maximization Problem** Find perturbation $\boldsymbol{\delta}$ to maximize loss (fool the model) under constraint (imperceptible): $\max_{\boldsymbol{\delta}}L(f(\boldsymbol{x}+\boldsymbol{\delta}),y)$ s.t. $\|\boldsymbol{\delta}\|_p\leq\varepsilon$ **White-box**: Model parameters/gradients known. **Black-box**: Only outputs known (transfer attack).

Black-Box Variants (Handwrite)

- **Score-based**: Can query continuous confidence scores (probabilities).
- **Decision-based**: Can only query discrete hard labels (0/1).

1. FGSM (Fast Gradient Sign Method) Constraint: ℓ_∞ norm (Change each pixel $\leq\varepsilon$). Linearize loss around $\boldsymbol{x}$ (1st order Taylor): $\boldsymbol{x}_{adv}=\boldsymbol{x}+\varepsilon\cdot\text{sign}(\nabla_{\boldsymbol{x}}L(\boldsymbol{\theta},\boldsymbol{x},y))$

- **Why Sign?** Optimal direction for ℓ_∞ constraint (corners of the hypercube). **One-step**: Fast but underfits (linear approximation).

2. PGD (Projected Gradient Descent) Iterative FGSM. More powerful, harder to defend. $\boldsymbol{x}_{t+1}=\Pi_{\boldsymbol{x}+\mathcal{S}}(\boldsymbol{x}_t+\alpha\cdot\text{sign}(\nabla_{\boldsymbol{x}}L))$

- **Project** (Π): Clip Π: range $[x-\varepsilon,x+\varepsilon]\cap[0,1]$.
- Universal 1st-order adversary.

3. ℓ_2 Attack (Gradient Direction) Total energy limited ($\|\boldsymbol{\delta}\|_2\leq\varepsilon$).

$\boldsymbol{\delta}=\varepsilon\cdot\frac{\nabla_{\boldsymbol{x}}L}{\|\nabla_{\boldsymbol{x}}L\|_2}$ (No sign function!)

Defense Adversarial Training:
Train on $\{\boldsymbol{x}_{adv},y\}$. $\min_{\boldsymbol{\theta}}\mathbb{E}_{(\boldsymbol{x},y)\sim D}[\max_\delta L(f_\theta(\boldsymbol{x}+\delta),y)]$.

Linear Model Exact Attack (Exam '21) Model $y=\boldsymbol{w}^T\boldsymbol{x}$, Loss $(y-\boldsymbol{w}^T(\boldsymbol{x}+\boldsymbol{\delta}))^2$, constraint $\|\boldsymbol{\delta}\|_\infty\leq\epsilon$.

- **Goal**: Maximize error. Make $\boldsymbol{w}^T\boldsymbol{\delta}$ as negative/positive as possible.
- **Optimal** $\boldsymbol{\delta}^*$: $\delta_i=-\epsilon\cdot\text{sign}(w_i)$ (if trying to decrease pred).
- **Max Loss**: $(y-\boldsymbol{w}^T\boldsymbol{x}-\underbrace{\boldsymbol{w}^T\boldsymbol{\delta}^*}_{-\epsilon\|\boldsymbol{w}\|_1})^2=(y-\boldsymbol{w}^T\boldsymbol{x}+\epsilon\|\boldsymbol{w}\|_1)^2$.

15. Matrix Factorization (RecSys)

Model Definition Approximate rating matrix $\boldsymbol{R}\approx\boldsymbol{P}\times\boldsymbol{Q}^T$. Prediction for user u, item i: $\hat r_{ui}=\boldsymbol{p}_u\cdot\boldsymbol{q}_i^T=\sum_{f=1}^k p_{u,f}\cdot q_{i,f}$ where $\boldsymbol{P}\in\mathbb{R}^{m\times k},\boldsymbol{Q}\in\mathbb{R}^{n\times k}$ (k: latent dim, $k\ll m,n$).

Objective Function (Regularized SE) Minimize loss J over observed set $\mathcal{K}$: $J=\sum_{(u,i)\in\mathcal{K}}(r_{ui}-\boldsymbol{p}_u\cdot\boldsymbol{q}_i^T)^2+\lambda(\|\boldsymbol{p}_u\|_2^2+\|\boldsymbol{q}_i\|_2^2)$

SGD Update Rules (Calculations) 1. Calculate Error: $e_{ui}=r_{ui}-\hat r_{ui}$
2. Update Factors: (Learning rate η), *Note*: Term $-\lambda\boldsymbol{p}_u$ is weight decay (prevents overfit). $\boldsymbol{p}_u\leftarrow\boldsymbol{p}_u+\eta(e_{ui}\boldsymbol{q}_i-\lambda\boldsymbol{p}_u)$; $\boldsymbol{q}_i\leftarrow\boldsymbol{q}_i+\eta(e_{ui}\boldsymbol{p}_u-\lambda\boldsymbol{q}_i)$

- **Biased MF**: $\hat r_{ui}=\mu+b_u+b_i+\boldsymbol{p}_u\cdot\boldsymbol{q}_i^T$.
- **ALS (Alternating Least Squares)**: Fix $\boldsymbol{P}$, solve $\boldsymbol{Q}$ (OLS); Fix $\boldsymbol{Q}$, solve $\boldsymbol{P}$.
- Good for *implicit feedback* & parallelization.

SVD vs. MF: **SVD**: Requires dense matrix (no missing). Slow $O(mn^2)$. **MF**: Handles **sparse** data. Fast $O(|\mathcal{K}|k)$. Approx.

16. Contrastive Learning (SimCLR)

Objective: **InfoNCE Loss** Maximize similarity of views from same image (pos); push away others (neg). $\mathcal{L}=-\log\frac{\exp(\text{sim}(z_i,z_j)/\tau)}{\sum_k\exp(\text{sim}(z_i,z_k)/\tau)}$

Key Components

- **Augmentations**: NOT random/all. **Composition** is key.
  - **Crop + Color Jitter**: Critical. Forces model to learn shape/texture, not color histograms.
- **Batch Size**: Needs large batch (more negatives).
- **Temperature** τ:
  - **Low** τ: Softmax becomes sharp. Model focuses heavily on **Hard Negatives** (difficult samples).
  - **High** τ: Gradients uniform across all negatives.

17. K-Nearest Neighbors (KNN)

- **Classification**: Majority vote of K neighbors. **Regression**: Average of K neighbors. **Hyperparameter** K:
  - **Small** K **(e.g., 1)**: High Variance, Low Bias. Complex boundary. Overfits (Train Error = 0).
  - **Large** K **(e.g.,** N**)**: Low Variance, High Bias. Simple boundary. Underfits (Predicts majority class).
- **Distance**: Sensitive to feature scaling. **Must standardize** data first!
- **Curse of Dimensionality**: As $D\uparrow$, all points become equidistant; Euclidean distance fails.

18. Advanced Exam Concepts (Tricks)

1. Convexity Operations (Crucial) Let f,g be convex functions.

- **Sum**: $\alpha f(x)+\beta g(x)$ is convex ($\forall\alpha,\beta\geq0$).
- **Max**: $h(x)=\max(f(x),g(x))$ is **Convex**.
- **Min**: $h(x)=\min(f(x),g(x))$ is **NOT** guaranteed convex.
- **Composition**: $f(g(x))$ convex if f convex non-decreasing & g convex.

2. Loss Minimizers

- Minimize MSE (L_2): $\sum(x_i-\theta)^2\implies\theta=\text{Mean}(x)$.
- Minimize MAE (L_1): $\sum|x_i-\theta|\implies\theta=\text{Median}(x)$.

3. ReLU Network Construction Identities

- **Identity**: $x=\text{ReLU}(x)-\text{ReLU}(-x)$. Weights: $\boldsymbol{w}=[1,-1]^T,\boldsymbol{v}=[1,-1]^T$. **Abs**: $|x|=\text{ReLU}(x)+\text{ReLU}(-x)$. Weights: $\boldsymbol{w}=[1,-1]^T,\boldsymbol{v}=[1,1]^T$. **Max**: $\max(a,b)=b+\text{ReLU}(a-b)$. $F(x)=\text{ReLU}(x_1-x_2)+x_2$.

4. Algorithm Nuances

- **K-Means++ Init**: Pick c_1 random. Sample next c with prob $P(x)\propto D(x)^2$ (Sq distance to nearest center). **K-Means Convergence**: J is **non-increasing** ($J_{t+1}\leq J_t$). Converges to local min. **EM Algorithm**: Likelihood $L(\theta)$ is **non-increasing** per iter. **Batch Norm (Test Mode)**: Use **global** running μ,σ (EMA from train), NOT batch stats.
- **GAN Optimal**: $D^*(x)=0.5$. If D is perfect (100% acc), gradients vanish $\to$ training fails.