Machine Learning & Deep Learning — Deep Study Notes

These notes go beyond definitions into mechanics, math, pitfalls, and interview-ready explanations. Use the quick answers at the end of each section to rehearse.

---

1) Learning Paradigms

1.1 Supervised vs Unsupervised vs Reinforcement Learning

Supervised - **Data**: Labeled `(x, y)` pairs. - **Objective**: Minimize expected loss $\mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(f_\theta(x), y)]$. - **Common losses**: MSE (regression), cross-entropy (classification), hinge (SVM). - **Examples**: Price prediction, disease diagnosis, sentiment classification. - **Key risks**: Overfitting, label leakage, distribution shift.

Unsupervised - **Data**: Unlabeled `x` only. - **Objectives**: Density estimation (e.g., VAEs), representation learning (e.g., PCA), clustering (e.g., K-Means), anomaly detection. - **Examples**: Customer segmentation, topic modeling, outlier flagging, compression. - **Key risks**: Non-identifiability, arbitrary cluster semantics, metric sensitivity.

Reinforcement Learning (RL) - **Setting**: Agent interacts with environment (MDP $\langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$). - **Objective**: Maximize expected discounted return $\mathbb{E}[\sum_t \gamma^t r_t]$. - **Methods**: Value-based (Q-learning), policy-based (REINFORCE), actor–critic, model-based RL. - **Examples**: Game playing (Go, Atari), robotics control, recommender systems (bandits). - **Key risks**: Exploration–exploitation trade-off, reward hacking, sample inefficiency, instability.

Interview quick answer: *Supervised learns from labels to predict outputs; unsupervised discovers structure without labels; RL learns a policy by trial-and-error to maximize long-term reward.*

---

2) Parametric vs Non-Parametric Models

Parametric - Fixed parameter count irrespective of data size. - Strong assumptions (e.g., linearity, Gaussian noise) → fast training/inference, lower variance, higher bias. - **Examples**: Linear/Logistic Regression, Naive Bayes, (fixed-architecture) Neural Nets are parametric (parameter count fixed), GLMs.

Non-Parametric - Effective complexity grows with data; fewer distributional assumptions → higher flexibility, higher variance, more data-hungry. - **Examples**: k-NN (stores data), Decision Trees, Random Forests, Gaussian Processes, Kernel Density Estimation.

Bayesian view: Parametric places priors over a fixed parameter vector; non-parametric places priors over function spaces (e.g., GPs with kernels) where capacity grows with data.

Interview quick answer: *Parametric: fixed-size model with stronger assumptions; non-parametric: capacity scales with data, fewer assumptions.*

---

3) Bias–Variance Trade-off

- **Decomposition (regression)**: $\mathbb{E}[(y - \hat{f}(x))^2] = \underbrace{\text{Bias}^2}_{(\mathbb{E}[\hat{f}]-f)^2} + \underbrace{\text{Variance}}_{\mathbb{E}[(\hat{f}-\mathbb{E}[\hat{f}])^2]} + \sigma^2_{\text{irreducible}}$.
- **High bias** → underfitting; **high variance** → overfitting.
- **Knobs**: Model capacity, regularization strength, data size, ensembling, feature noise.

Interview quick answer: *Bias is error from wrong assumptions; variance is error from sensitivity to data. We balance them via capacity and regularization.*

---

4) Overfitting vs Underfitting

Underfitting: Train and test errors both high (model too simple). Fix: add features, use richer model, reduce regularization.

Overfitting: Train error low, test error high (memorization/noise fitting). Fix: add regularization (L1/L2/dropout), early stopping, data augmentation, cross-validation, ensembling, simplify model, collect more data.

Detection: Learning curves (error vs. n_samples), gap between train/val metrics, high variance across folds, unstable predictions.

Data leakage red flags: Features computed using test-time information, target encoded without proper CV, temporal leakage.

---

5) Regularization (with L1 vs L2)

Purpose: Penalize complexity to improve generalization.

- **L2 (Ridge)**: Add $\lambda\|w\|_2^2$ → closed-form for linear models: $\hat{w} = (X^\top X + \lambda I)^{-1} X^\top y$. Shrinks coefficients smoothly; correlated features get similar weights; stabilizes ill-conditioned problems.
- **L1 (Lasso)**: Add $\lambda\|w\|_1$ → promotes sparsity via soft-thresholding; performs implicit feature selection.
- **Elastic Net**: $\lambda(\alpha\|w\|_1 + (1-\alpha)\|w\|_2^2)$: balances sparsity and grouping effect.

Bayesian interpretation: L2 ≡ Gaussian prior on weights; L1 ≡ Laplace prior.

Geometry: L1 constraint (diamond) causes corner solutions at axes → zeros; L2 (circle) rarely touches axes.

When to use: L1 for feature selection/high-dimensional sparse signals; L2 for multicollinearity/numerical stability; Elastic Net when p ≫ n with correlated groups.

---

6) Decision Trees

Core idea: Recursively partition feature space to minimize impurity.

- **Impurity**:
- *Gini*: $G = \sum_k p_k(1-p_k) = 1 - \sum_k p_k^2$
- *Entropy*: $H = -\sum_k p_k \log p_k$
- *Regression*: minimize variance / MSE.
- **Split selection**: For each feature & threshold, compute impurity decrease $\Delta = I(parent) - \sum_{child} w_c I(child)$.
- **Stopping/Pruning**: Max depth, min samples per split/leaf, cost-complexity pruning (penalty $\alpha \cdot \#leaves$).

Pros: Interpretable, handles nonlinearity & interactions, little preprocessing, works with mixed feature types, missing-value surrogates.

Cons: High variance/overfitting, axis-aligned splits, unstable to small perturbations, biased towards features with many splits unless corrected.

Best practices: Use pruning, control depth, or prefer ensembles (RF/GBTs) for accuracy.

---

7) Random Forests vs Gradient Boosting

Random Forest (Bagging) - Train many deep, decorrelated trees on bootstrapped samples; at each split use a random subset of features (mtry). - **Prediction**: Average (regression) / majority vote (classification). - **Strengths**: Robust, good out-of-the-box, OOB error estimate, handles high-dimensional data, less tuning. - **Weaknesses**: Larger models, slower inference than single tree, can underperform GBTs on tabular data with careful tuning.

Gradient Boosting (e.g., XGBoost, LightGBM, CatBoost) - Sequentially add shallow trees fit to gradients of the loss (functional gradient descent). - **Key knobs**: learning_rate, n_estimators, max_depth/num_leaves, subsample, colsample, regularization (ℓ_1/ℓ_2), min_child_weight. - **Strengths**: State-of-the-art on many tabular tasks, handles heterogeneity, flexible losses. - **Weaknesses**: Sensitive to hyperparameters, risk of overfitting, less parallelizable.

When to choose: Start with RF when you want stability/low tuning; use GBTs when you can tune and need top accuracy.

---

8) Support Vector Machines (SVM)

Maximum margin classifier: Find hyperplane $w^\top x + b = 0$ maximizing margin $2/\|w\|$ while classifying data with slack variables ξ_i .

Primal (soft-margin):

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_i \xi_i \quad \text{s.t.} \quad y_i(w^\top x_i + b) \geq 1 - \xi_i,\ \xi_i \geq 0$$

Dual leads to **kernel trick** via $K(x_i, x_j) = \phi(x_i)^\top \phi(x_j)$, enabling nonlinear decision boundaries.

Common kernels: Linear, Polynomial, RBF/Gaussian $\exp(-\gamma\|x - x'\|^2)$, Sigmoid.

Pros: Effective in high dimensions, robust margins, sparse support vectors. **Cons**: Scaling to millions of samples is hard; kernel/C, γ tuning required; less interpretable.

---

9) K-Means Clustering

Objective: Minimize within-cluster SSE $\sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$.

Algorithm (Lloyd's) 1) Initialize centroids (k-means++ preferred). 2) Assign step: $c(x) = \arg\min_i \|x - \mu_i\|^2$. 3) Update step: $\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$. 4) Repeat until assignments stop changing / inertia converges.

Limitations: Requires K; assumes spherical/equal-variance clusters; sensitive to scale & outliers; only Euclidean metric; local minima.

Diagnostics: Elbow plot, silhouette score, Davies–Bouldin, stability across seeds.

---

10) PCA (Principal Component Analysis)

Goal: Orthogonal linear projections capturing maximal variance.

Procedure 1) Standardize features (zero mean, unit variance). 2) Covariance matrix $\Sigma = \frac{1}{n-1} X^\top X$ (after centering). 3) Eigendecompose $\Sigma = V \Lambda V^\top$. 4) PCs are columns of V ; explained variance ratios = diag(Λ)/ trace(Λ). 5) Project: $Z = X V_k$.

Interpretation: PCA directions = eigenvectors; scores = coordinates in PC space; loadings = feature contributions.

When to use: Visualization (2D/3D), denoising, collinearity removal, speedups for downstream models; not ideal when features are not linearly related.

Pitfalls: Scaling required; PCs are not guaranteed interpretable; sensitive to outliers; for mixed types, consider MCA/FAMD.

---

11) Correlation vs Covariance

- **Covariance**: $\mathrm{Cov}(X, Y) = \mathbb{E}[(X - \mu_X)(Y - \mu_Y)]$, units depend on X and Y; magnitude not standardized.
- **Correlation**: $\rho = \frac{\mathrm{Cov}(X,Y)}{\sigma_X \sigma_Y} \in$ [−1,1]; unitless and comparable.
- **Pearson vs Spearman**: Pearson measures linear relation; Spearman measures rank/monotonic relation; Kendall τ for concordance.

---

12) Probability Distributions in ML

- **Gaussian**: ubiquitous due to CLT; linear regression assumes Gaussian i.i.d. errors for exact inference and CI; many algorithms rely on Gaussian priors/noise.
- **Bernoulli/Binomial**: classification labels and count successes.
- **Poisson**: counts/events in fixed interval; GLM via log link.
- **Exponential/Gamma**: waiting times.
- **Categorical/Multinomial**: multi-class labels.

Why assumptions matter: E.g., Naive Bayes with Gaussian likelihoods for continuous features; violations degrade calibration and inference quality.

---

13) Cross-Validation

- **Hold-out**: simple but high variance.
- **k-Fold (typical k=5 or 10)**: average metrics across folds; stratify for class imbalance.
- **LOOCV**: low bias, high variance, expensive.
- **Time-series CV**: rolling/expanding windows (never shuffle across time).
- **Nested CV**: outer loop for evaluation, inner for hyperparameter tuning (prevents optimistic bias).

Best practices: Preserve groups (GroupKFold), prevent leakage in pipelines (fit scalers/encoders within each fold), use consistent seeds.

---

14) Classification Metrics

Let TP, FP, TN, FN be confusion matrix entries. - **Accuracy** = (TP+TN)/(TP+FP+TN+FN) — misleading with imbalance. - **Precision** = TP/(TP+FP) — purity of positive predictions. - **Recall (TPR)** = TP/(TP+FN) — coverage of actual positives. - **F1** = 2·(Precision·Recall)/(Precision+Recall) — balance; use when uneven class costs. - **ROC Curve**: plot TPR vs FPR over thresholds; **AUC** = probability a random positive scores higher than a random negative. - **PR Curve**: precision vs recall — better for heavy imbalance; **AUPRC** more informative than ROC in rare-event settings. - **Calibration**: Brier score, reliability diagrams; crucial for decision thresholds and risk.

---

15) Backpropagation

Goal: Compute $\nabla_\theta \mathcal{L}$ efficiently via chain rule on computational graph.

For layer: $a^{(l)} = \sigma(W^{(l)} h^{(l-1)} + b^{(l)})$, with loss $\mathcal{L}$. - **Backward**: $\delta^{(l)} = (W^{(l+1)})^\top \delta^{(l+1)} \odot \sigma'(z^{(l)})$. - **Gradients**: $\partial\mathcal{L}/\partial W^{(l)} = \delta^{(l)} (h^{(l-1)})^\top$, $\partial\mathcal{L}/\partial b^{(l)} = \delta^{(l)}$.

Optimizers: SGD, Momentum, Nesterov, RMSProp, Adam/AdamW; learning-rate schedules (cosine, step, warmup).

---

16) Vanishing/Exploding Gradients

- **Why**: Repeated multiplication by Jacobians with singular values <1 (vanish) or >1 (explode); severe with sigmoids/tanh and deep/recurrent stacks.
- **Mitigation**: ReLU/LeakyReLU/GELU, careful init (He/Xavier), batch/layer norm, residual/skip connections, gated RNNs (LSTM/GRU), gradient clipping, shorter unrolls.

---

17) CNNs vs RNNs

CNNs - Convolutions + pooling extract local spatial features; weight sharing → translation equivariance; deep stacks → hierarchies (edges→textures→objects). - **Use cases**: Vision (classification, detection, segmentation), time series (with 1D convs), audio spectrograms.

RNNs (incl. LSTM/GRU) - Process sequences via hidden state; capture temporal dependencies. - **Use cases**: Language modeling, speech, time-series forecasting.

Modern note: Transformers (self-attention) often replace RNNs for long-range dependencies; CNNs remain strong in vision, often hybridized with attention.

---

18) Dropout

- **Mechanism**: During training, randomly zero units with probability p; scale activations at inference (or inverted dropout scales at train time by 1/(1-p)).
- **Effect**: Acts as model averaging over thinned networks; reduces co-adaptation and overfitting.
- **When**: Fully-connected layers; in CNNs often use spatial dropout or rely on batch-norm/ augmentations.

---

19) Debugging: Train OK, Production Bad

Checklist 1) **Data/Feature Drift**: Compare distributions (KS test, PSI), monitor missingness and categorical cardinality. 2) **Training–Serving Skew**: Ensure identical preprocessing (scaling, encoding, imputation) and order of features; serialize pipelines. 3) **Concept Drift**: Labels or relationships change; implement periodic re-training or online learning. 4) **Leakage in training**: Rebuild with strict CV and time-aware splits. 5) **Environment parity**: Library versions, numerical precision, model file corruption, locale issues. 6) **Monitoring**: Latency, input ranges, prediction histograms, calibration drift, business KPIs. 7) **Shadow/Canary**: Route fraction of traffic, compare against incumbent.

Tactics: Add input validators, feature attributions (SHAP) to spot broken features, build golden test cases.

---

20) Imbalanced Datasets

Symptoms: High accuracy but poor recall on minority; ROC-AUC looks fine while PR-AUC is low.

Techniques - **Data-level**: Random under/over-sampling; **SMOTE/Borderline-SMOTE/ADASYN**; cluster-based under-sampling. - **Algorithm-level**: Class weights / cost-sensitive losses (weighted cross-entropy, focal loss), threshold moving, anomaly detection framing. - **Evaluation**: Use stratified CV, PR-AUC, class-wise recall/precision, confusion matrix; set operating point based on costs. - **Pipeline hygiene**: Apply resampling **inside** CV folds only to avoid leakage; maintain separations in time-series.

Focal loss (for hard examples): $\mathcal{L} = -\alpha(1-p_t)^\gamma \log p_t$.

Worked Micro-Examples (Interview Ready)

Overfitting detection: Train acc 99%, Val acc 85% → increase L2, add dropout/augmentations, early stop; check leakage.

Choosing K in K-Means: Run k ∈ {2..10}, plot elbow; also inspect silhouette score and cluster stability across seeds; if elongated clusters, consider GMM/DBSCAN.

SVM kernel selection: Start linear when d≫n (text); otherwise RBF with CV over C and γ; standardize features.

PCA intuition line: *PCA rotates the coordinate system to align axes with directions of maximal variance; we keep the top k axes to compress with minimum reconstruction error.*

Regularization one-liner: *L1 drives sparsity (feature selection); L2 stabilizes and shares weight among correlated predictors.*

Formulas & Snippets

- **Cross-entropy** (binary): $\mathcal{L} = -[y \log p + (1-y)\log(1-p)]$.
- **Softmax**: $\mathrm{softmax}(z_i) = e^{z_i} / \sum_j e^{z_j}$.
- **Information Gain**: $IG(S, A) = H(S) - \sum_v \frac{|S_v|}{|S|} H(S_v)$.
- **Gini decrease** analogous with G instead of H.
- **Ridge closed form**: above; **Lasso** solved via coordinate descent/ISTA/FISTA.
- **k-NN**: majority vote among k nearest (Euclidean/other); sensitive to scale → standardize; KD-tree/ball tree for speed.

Common Pitfalls & Remedies

- **Data leakage**: Fit scalers/encoders on full data before split → *Wrong.* Always fit inside CV folds.
- **Inconsistent preprocessing**: Different category mappings between train/serve → serialize pipeline artifacts.
- **Imbalance**: Reporting accuracy only → include PR-AUC, recall@k.
- **Time series**: Random KFold → use time-based splits.
- **PCA misuse**: Applying PCA before train/test split using whole dataset → leakage.

Practice Checklists

Supervised vs Unsupervised vs RL - Define; give 2 examples each; state objective function.

Parametric vs Non-Parametric - Define; give 2 examples; Bayesian angle.

Bias/Variance & Over/Underfitting - Define; diagnose via curves; list 3 prevention methods.

Regularization - State penalties, geometry, Bayesian priors; when to use L1 vs L2.

Decision Trees / RF / GBT - Impurity measures; pruning; ensemble differences & trade-offs.

SVM - Margin concept; primal/dual; kernel trick; when linear vs RBF.

K-Means - Objective; algorithm; picking K; limitations & alternatives.

PCA - Steps; variance explained; when not to use.

Metrics - Confusion matrix; pick metrics for imbalance.

Deep Learning - Backprop equations; vanishing gradients & fixes; CNN vs RNN vs Transformer; dropout purpose.

Production - Drift detection; monitoring; shadow testing; retraining policy.

Suggested Mini-Projects for Mastery

1) **Tabular classification**: Train LR, SVM, RF, XGBoost; compare CV metrics, calibration, SHAP plots. 2) **Clustering**: Apply K-Means/DBSCAN on a customer dataset; evaluate with silhouette and business labels. 3) **Dimensionality reduction**: PCA vs UMAP on high-dim data; visualize; train downstream classifier. 4) **Imbalance**: Fraud dataset with SMOTE vs class weights vs focal loss; compare PR-AUC. 5) **Drift monitoring**: Build a simple PSI/KS dashboard and alert thresholds.

One-Minute Answers (for rapid recall)

- *Supervised/Unsupervised/RL*: labels vs structure vs reward-maximizing interaction.
- *Parametric/Non-parametric*: fixed-params & assumptions vs data-growing flexibility.
- *Bias/Variance*: wrong assumptions vs sensitivity; balance with capacity/regularization.
- *Over/Underfit*: high train–test gap vs both bad; fix with reg/early stop/augmentations.
- *L1 vs L2*: sparsity/selection vs shrinkage/stability; Laplace vs Gaussian priors.
- *Trees*: greedy impurity minimization; prune; ensembles improve stability.
- *RF vs GBT*: bagged deep trees vs boosted shallow trees; stability vs top accuracy.
- *SVM*: maximum margin; kernel trick for nonlinearity.
- *K-Means*: minimize within-cluster SSE; needs K; spherical clusters.
- *PCA*: orthogonal directions of max variance; good for compression/denoising.
- *Metrics*: F1/PR-AUC for imbalance; calibrate probabilities.
- *Backprop*: chain rule over graph; use AdamW and proper init.
- *Vanishing gradients*: ReLU, residuals, normalization.
- *CNN vs RNN*: spatial vs temporal; Transformers for long context.
- *Dropout*: random unit dropping as regularizer.
- *Prod debug*: drift, skew, leakage; monitor & canary.
- *Imbalance*: resample, class weights, focal loss, PR-AUC.*