Advanced Machine Learning

Silvan Stadelmann - 2. Februar 2026 - v0.1.0

github.com/silvasta/summary-aml

Current font size: 10.95pt

1 Key ML Concepts

1.1 What is Machine Learning?

ML enables systems to learn from data without explicit programming. Goals: Prediction, inference, decision-making.

Types of ML:

- **Supervised**: Labeled data; predict y from x (e.g., regression, classification).
- **Unsupervised**: Unlabeled data; find patterns (e.g., clustering, dimensionality reduction).
- **Reinforcement**: Agent learns via rewards (e.g., MDPs).
- **Semi-supervised/Self-supervised**: Mix of labeled/unlabeled.

ML Pipeline: Data collection $\to$ Preprocessing $\to$ Feature engineering $\to$ Model selection $\to$ Training $\to$ Evaluation $\to$ Deployment.

1.2 Empirical Risk Minimization (ERM)

Core principle: Minimize average loss on training data as proxy for true risk.

Formulas:

- True Risk: $R(f) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(f(x),y)]$, where ℓ is loss function, $\mathcal{D}$ is data distribution.
- Empirical Risk: $\hat{R}(f) = \frac{1}{n}\sum_{i=1}^{n}\ell(f(x_i),y_i)$.
- Goal: $\hat{f} = \text{argmin}_f\, \hat{R}(f)$ (often with regularization).

Trick: Overfitting occurs when $\hat{R} \ll R$; use validation sets, cross-validation.

1.3 Bias-Variance Tradeoff

Decomposes generalization error: $\mathbb{E}[(y-\hat{y})^2] = \text{Bias}^2 + \text{Variance} + \text{Noise}$.

Formulas:

- Bias: $\mathbb{E}[\hat{y}] - y$ (systematic error).
- Variance: $\mathbb{E}[(\hat{y} - \mathbb{E}[\hat{y}])^2]$ (sensitivity to data).

Exam Hint: High bias $\to$ underfitting (simple models); high variance $\to$ overfitting (complex models). Regularization reduces variance.

1.4 Basic Loss Functions

Common in early exams for supervised tasks.

- Regression (MSE): $\ell(y,\hat{y}) = (y-\hat{y})^2$.
- Classification (0-1 Loss): $\ell(y,\hat{y}) = \mathbb{I}(y \neq \hat{y})$.
- Logistic Loss: $\ell(y,p) = -y\log p - (1-y)\log(1-p)$ (for binary).

Trick: For optimization, use surrogates (e.g., hinge loss for SVM instead of 0-1).

1.5 Overfitting & Regularization

Prevent by adding penalty: $\hat{f} = \text{argmin}_f\, \hat{R}(f) + \lambda\Omega(f)$, where Ω is complexity (e.g., L2: $\|w\|^2$).

Hint: Cross-validation for λ; exams may ask to derive regularized ERM.

2 Representations

2.1 Key Concepts & Tricks

- **Representations**: Feature maps $\phi : \mathcal{X} \to \mathbb{R}^d$ for non-linear models (e.g., kernels). Exam trick: Use for bounds in estimation (e.g., info in embedded spaces).
- **CRLB**: Lower bound on variance of unbiased estimators $\hat{\theta}$ of param θ. Assumes regularity: differentiable log-likelihood, finite variance.
- Trick: CRLB achieved iff estimator is efficient (e.g., MLE in exponential families). Multivariate: Use inverse Fisher matrix.
- Hint for exams: Always check unbiasedness first; compute via Hessian or score function.

2.2 Formulas: Fisher Information

Fisher info measures param sensitivity in likelihood. For scalar θ:

$$I(\theta) = \mathbb{E}\left[\left(\frac{\partial}{\partial\theta}\log p(X|\theta)\right)^2\right] = -\mathbb{E}\left[\frac{\partial^2}{\partial\theta^2}\log p(X|\theta)\right]$$

Multivariate ($\theta \in \mathbb{R}^k$): Matrix form

$$[I(\theta)]_{ij} = \mathbb{E}\left[\frac{\partial\log p}{\partial\theta_i}\frac{\partial\log p}{\partial\theta_j}\right] = -\mathbb{E}\left[\frac{\partial^2\log p}{\partial\theta_i\partial\theta_j}\right]$$

Trick: For iid samples $X_1,\dots,X_n$, $I_n(\theta) = nI(\theta)$.

Example: Gaussian $\mathcal{N}(\mu,\sigma^2=1)$

Score: $\frac{\partial\log p}{\partial\mu} = x-\mu$. Fisher: $I(\mu) = 1$.

2.3 Rao-Cramér Lower Bound (CRLB)

For unbiased estimator $\hat{\theta}(X)$:

$$\text{Var}(\hat{\theta}) \geq \frac{1}{I(\theta)} \quad \text{(scalar)}$$

Multivariate (for function $g(\theta)$):

$$\text{Var}(g(\hat{\theta})) \geq \left(\frac{\partial g}{\partial\theta}\right)^T I(\theta)^{-1}\left(\frac{\partial g}{\partial\theta}\right)$$

General CRLB:

$$\text{Cov}(\hat{\theta}) \succeq I(\theta)^{-1}$$

Trick: Equality if $\hat{\theta} = a(\theta)\cdot s(X) + b(\theta)$, where $s(X)$ is sufficient statistic (e.g., in Gaussians, sample mean achieves CRLB).

2.4 Calculus Recipes & Derivations

- Compute $I(\theta)$: (1) Write $\log L(\theta|X) = \sum\log p(x_i|\theta)$. (2) Take 2nd deriv or score sq. (3) Expectation over $p(X|\theta)$.
- For representations: Info in feature space: $I_\phi(\theta) = \mathbb{E}[\phi(X)^T\phi(X)]^{-1}$ (e.g., for linear models).
- Exam hint: In ML, CRLB bounds learning rates (e.g., variance in param est. for neural nets).

Regularity Conditions: Support indep. of θ; $\int p(x|\theta)dx = 1$ differentiable under integral.

3 Gaussian Processes (GPs)

Definition: GP is a distribution over functions $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x},\mathbf{x}'))$, where $m(\cdot)$ is mean function (often 0), $k(\cdot,\cdot)$ is kernel (covariance function).

Key Kernels:

- RBF: $k(\mathbf{x},\mathbf{x}') = \sigma_f^2\exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\ell^2}\right)$ (lengthscale ℓ, variance σ_f^2).

- Linear: $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T\mathbf{x}' + c$.
- Matérn: $k(r) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}r}{\ell}\right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}r}{\ell}\right)$ ($\nu = 3/2$ or $5/2$ for smoothness).

GP Regression (Noisy Observations): $y = f(\mathbf{x}) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$. Training data: $\mathbf{X}, \mathbf{y}$.

Prior: $\mathbf{f} \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$, where $K_{ij} = k(x_i, x_j)$.

Posterior predictive: For test points $\mathbf{X}_*$, $\mathbf{f}_*|\mathbf{y} \sim \mathcal{N}(\bar{\mathbf{f}}_*, \text{cov}(\mathbf{f}_*))$,

$$\bar{\mathbf{f}}_* = \mathbf{K}_{*\mathbf{X}}(\mathbf{K}_{\mathbf{XX}} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y},$$
$$\text{cov}(\mathbf{f}_*) = \mathbf{K}_{**} - \mathbf{K}_{*\mathbf{X}}(\mathbf{K}_{\mathbf{XX}} + \sigma_n^2\mathbf{I})^{-1}\mathbf{K}_{\mathbf{X}*}.$$

Marginal Likelihood (for hyperparams θ): $\log p(\mathbf{y}|\mathbf{X}, \theta) = -\frac{1}{2}\mathbf{y}^T(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K} + \sigma_n^2\mathbf{I}| - \frac{n}{2}\log 2\pi$.

Tricks/Exam Hints:

- Cholesky decomp for inversion: Stable for positive-definite $\mathbf{K}$.
- Optimize θ via gradient descent on log-marginal (derivs w.r.t. ℓ, σ_f).
- GPs for classification: Use logistic/sigmoid + Laplace approx or EP.
- Scalability: Sparse GPs (e.g., FITC) approx large datasets.

4 Ensembles

Key Methods:

- **Bagging** (Bootstrap Aggregating): Train M models on bootstrap samples, average predictions. Reduces variance.
- **Random Forest**: Bagging + random feature subsets at splits. Importance: $I(f) = \sum_{\text{nodes}} \Delta\text{impurity} \cdot p(\text{node})$.
- **Boosting** (e.g., AdaBoost): Sequential, weight misclassified points. Final: $H(\mathbf{x}) = \text{sign}\left(\sum_m \alpha_m h_m(\mathbf{x})\right)$, $\alpha_m = \frac{1}{2}\log\frac{1-\epsilon_m}{\epsilon_m}$.
- **Gradient Boosting**: Minimize loss $L = \sum_i l(y_i, F(\mathbf{x}_i))$, update $F_m = F_{m-1} + \nu h_m$, where h_m fits pseudo-residuals $r_{im} = -\frac{\partial l}{\partial F_{m-1}(\mathbf{x}_i)}$.

Bias-Variance Tradeoff: Ensembles reduce variance (bagging) or bias (boosting). Error bound for AdaBoost: $\epsilon \leq 2^M \prod_m \sqrt{\epsilon_m(1-\epsilon_m)}$.

Tricks/Exam Hints:

- Diversity: Key to ensembles; measure via correlation of base learners.
- Overfitting: Boosting can overfit; use early stopping or shrinkage $\nu < 1$.
- For proofs: Derive exponential loss minimization for AdaBoost weights.

5 Support Vector Machines (SVMs)

5.1 Hard-Margin SVM (Linearly Separable)

Primal: Minimize $\frac{1}{2}\|\mathbf{w}\|^2$ s.t. $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1\ \forall i$.

Dual: Maximize $\sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$ s.t. $\alpha_i \geq 0$, $\sum_i \alpha_i y_i = 0$.

Decision: $f(\mathbf{x}) = \text{sign}\left(\sum_i \alpha_i y_i \mathbf{x}_i^T\mathbf{x} + b\right)$. Margin: $\gamma = \frac{2}{\|\mathbf{w}\|}$.

Support vectors: Points where $\alpha_i > 0$ (on margin).

5.2 Soft-Margin SVM

Primal: Minimize $\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i$ s.t. $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$, $\xi_i \geq 0$.

Hinge loss: $\ell(y, \hat{y}) = \max(0, 1 - y\hat{y})$.

Dual: Same as hard-margin but $0 \leq \alpha_i \leq C$.

Trick: C trades bias/variance; large C ▯ hard-margin.

5.3 Kernel Trick

Replace $\mathbf{x}_i^T\mathbf{x}_j$ with $k(\mathbf{x}_i, \mathbf{x}_j)$. Dual becomes: Maximize $\sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$.

Common kernels:

- Linear: $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T\mathbf{z}$
- Polynomial: $k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T\mathbf{z} + c)^d$
- RBF: $k(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{z}\|^2}{2\sigma^2}\right)$

Mercer's condition: Kernel matrix positive semi-definite.

6 Ensemble Methods

6.1 Bagging (Bootstrap Aggregating)

Aggregate B bootstrapped models: $\hat{f}(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}_b(\mathbf{x})$ (regression) or majority vote (classification).

Reduces variance: Var($\hat{f}$) ≈ $\frac{1}{B}$ Var(single model) if uncorrelated.

Random Forest: Bagging + random feature subsets at splits. OOB error for validation.

6.2 Boosting

Sequential: Train weak learners on reweighted data.

AdaBoost: Weights $w_i^{(t+1)} = w_i^{(t)}\exp(-\alpha_t y_i h_t(\mathbf{x}_i))$, normalized. $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$, where ϵ_t = weighted error.

Final: $H(\mathbf{x}) = \text{sign}\left(\sum_t \alpha_t h_t(\mathbf{x})\right)$.

Gradient Boosting: Minimize loss by adding trees fitting residuals. Update: $f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \nu \cdot h_m(\mathbf{x})$ (ν: shrinkage).

Trick: Boosting reduces bias; risk of overfitting—use early stopping.

6.3 Key Tricks & Comparisons

Bias-variance: Ensembles ▯ variance (bagging) or ▯ bias (boosting).

Error bound (AdaBoost): Training error ≤ $\exp(-2\sum_t(\frac{1}{2} - \epsilon_t)^2)$.

Exam hint: For non-separable data, use soft-margin or kernels; ensembles for high-variance base learners (e.g., deep trees).

7 Neural Networks: Basics

MLP Forward Pass: For layer l, $\mathbf{z}^{(l)} = \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$, $\mathbf{a}^{(l)} = \sigma(\mathbf{z}^{(l)})$.

Activation Functions: - Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$, $\sigma'(x) = \sigma(x)(1-\sigma(x))$. - ReLU: $\sigma(x) = \max(0, x)$, derivative 1 if $x > 0$ else 0. *Trick*: Mitigates vanishing gradients. - Softmax: $\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$, for classification.

Loss Functions: - MSE: $\mathcal{L} = \frac{1}{2N}\sum_i(\hat{y}_i - y_i)^2$. - Cross-Entropy: $\mathcal{L} = -\sum_i y_i \log(\hat{y}_i)$. *Hint*: For multi-class, combine with softmax.

Backpropagation: - Output gradient: $\delta^{(L)} = \frac{\partial\mathcal{L}}{\partial\mathbf{z}^{(L)}} = (\hat{\mathbf{y}} - \mathbf{y}) \odot \sigma'(\mathbf{z}^{(L)})$. - Hidden: $\delta^{(l)} = (\mathbf{W}^{(l+1)T}\delta^{(l+1)}) \odot \sigma'(\mathbf{z}^{(l)})$. - Weight update: $\frac{\partial\mathcal{L}}{\partial\mathbf{W}^{(l)}} = \delta^{(l)}\mathbf{a}^{(l-1)T}$. *Trick*: Use chain rule; initialize weights $\sim \mathcal{N}(0, \frac{2}{n_{in}})$ (He init for ReLU).

Optimization Tricks: Gradient Descent: $\theta \leftarrow \theta - \eta\nabla\mathcal{L}$. Momentum: Add velocity term. Adam: Adaptive learning rates with moments.

8 Attention Mechanisms

Scaled Dot-Product Attention: Attention$(\mathbf{Q}, \mathbf{K}, \mathbf{V})$ = softmax $\left(\frac{\mathbf{QK}^T}{\sqrt{d_k}}\right)\mathbf{V}$. - $\mathbf{Q}$: Queries $(n \times d_k)$, $\mathbf{K}$: Keys $(m \times d_k)$, $\mathbf{V}$: Values $(m \times d_v)$. - *Trick*: Scaling prevents softmax saturation; causal mask for decoders (upper triangle $-\infty$).

Multi-Head Attention: MultiHead = Concat(head$_1, \ldots,$ head$_h$)$\mathbf{W}^O$, - Each head: head$_i$ = Attention$(\mathbf{QW}_i^Q, \mathbf{KW}_i^K, \mathbf{VW}_i^V)$. - *Hint*: $h = 8$ typical; allows parallel focus on subspaces.

Self-Attention: $\mathbf{Q} = \mathbf{K} = \mathbf{V} = \mathbf{XW}$ (input projection). *Exam Tip*: Captures dependencies without recurrence; $O(n^2)$ time.

9 Transformers

Architecture: Encoder (self-attn + FFN) stack; Decoder (masked self-attn + enc-dec attn + FFN) stack. - FFN: Two linear layers with ReLU: FFN$(\mathbf{x}) = \max(0, \mathbf{xW}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2$. -

Residual: $\mathbf{x} \leftarrow \mathbf{x} + \text{Sublayer}(\mathbf{x})$. LayerNorm after.

Positional Encoding: $\text{PE}_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right)$, $\text{PE}_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right)$. - Added to input embeddings. *Trick*: Allows order awareness; fixed or learned.

Training/Inference Tricks: Teacher forcing for training; beam search for generation. *Exam Focus*: Derive attention gradients or compare to RNNs (transformers handle long-range better).

10 Computer Vision

10.1 Convolutional Neural Networks (CNNs)

Key Concepts: Parameter sharing, local connectivity, translation invariance. Architectures: LeNet (simple), AlexNet (deep with ReLU/dropout).

Discrete Convolution (2D): For input I (size $H \times W \times C$), kernel K (size $k_h \times k_w \times C$),

$$O[i,j] = \sum_{m=0}^{k_h-1}\sum_{n=0}^{k_w-1}\sum_{c=1}^{C} I[i+m, j+n, c] \cdot K[m,n,c] + b$$

Output size: $\lfloor (H - k_h + 2p)/s \rfloor + 1$, where $p =$ padding, $s =$ stride.

Pooling (Max/Avg): Reduces dims, e.g., max-pool: $O[i,j] = \max_{m,n} I[i \cdot s + m, j \cdot s + n]$.

Activation Functions: ReLU: $f(x) = \max(0,x)$; Softmax for classification: $\sigma(z_i) = e^{z_i}/\sum e^{z_j}$.

10.2 Training & Optimization

Loss Functions: Cross-entropy for multi-class: $\mathcal{L} = -\sum y_i \log \hat{y}_i$.

Backpropagation in CNNs: Gradients via chain rule. For conv layer: - Weight grad: $\frac{\partial \mathcal{L}}{\partial K[m,n,c]} = \sum_{i,j} \frac{\partial \mathcal{L}}{\partial O[i,j]} \cdot I[i+m, j+n, c]$. - Input grad: Rotate kernel 180° and convolve with output grad.

Tricks: Batch norm: Normalize activations $\hat{x} = (x - \mu)/\sqrt{\sigma^2 + \epsilon}$, then $\gamma \hat{x} + \beta$. Dropout: Randomly zero neurons during training (prob p).

10.3 Applications & Metrics

Classification: Output via FC layer + softmax.

Object Detection Basics: Bounding boxes; IoU: $\text{IoU} = \frac{\text{Area}(A \cap B)}{\text{Area}(A \cup B)}$.

Segmentation: Pixel-wise classification; U-Net: Encoder-decoder with skip connections.

Exam Hints: Derive output shapes; explain why CNNs > FC nets (fewer params: $O(k^2C)$ vs. $O(HWC)$).

11 Graph Neural Networks (GNNs)

11.1 Basics & Notation

Graph $G = (V, E)$, $|V| = n$ nodes, adjacency matrix $\mathbf{A} \in \{0,1\}^{n \times n}$ (symmetric for undirected). Feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ (node features). Degree matrix $\mathbf{D} = \text{diad}(\sum_j A_{ij})$.

Normalized adjacency: $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ (self-loops), $\hat{\mathbf{A}} = \mathbf{D}^{-1/2}\tilde{\mathbf{A}}\mathbf{D}^{-1/2}$ (symmetric normalization).

Message passing: Update node v as $h_v^{(l+1)} = \sigma\left(\sum_{u \in \mathcal{N}(v)} m_{u \to v}^{(l)}\right)$, where m aggregates neighbor info.

11.2 Graph Convolutional Network (GCN)

Layer: $\mathbf{H}^{(l+1)} = \sigma(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)})$, with $\mathbf{H}^{(0)} = \mathbf{X}$.

Spectral view: Approximation of graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$, normalized $\hat{\mathbf{L}} = \mathbf{I} - \hat{\mathbf{A}}$.

Over-smoothing: Deep GCNs make representations similar; mitigate with residual connections or normalization.

11.3 Graph Attention Network (GAT)

Attention: $\alpha_{ij} = \text{softmax}_j\left(\text{LeakyReLU}\left(\mathbf{a}^\top[\mathbf{W}h_i \| \mathbf{W}h_j]\right)\right)$.

Update: $h_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i) \cup i} \alpha_{ij}\mathbf{W}h_j^{(l)}\right)$. Multi-head: Concat or average heads.

11.4 Tasks & Pooling

Node classification: Predict labels from node embeddings.

Graph classification: Pool via readout $r = f(\{h_v | v \in V\})$ (e.g., mean, max, sum). Hierarchical pooling (e.g., DiffPool).

Link prediction: Score edges with $s(u,v) = h_u^\top h_v$ or MLP on concatenated embeddings.

Tricks: Use skip connections for deep GNNs; normalize features; handle heterophily with signed messages or higher-order neighbors.

12 Information Theory

12.1 Key Measures

Entropy: $H(X) = -\mathbb{E}_{p(x)}[\log p(x)] = -\sum p(x) \log p(x)$ (discrete); continuous: $-\int p(x) \log p(x)\, dx$.

Joint entropy: $H(X,Y) = -\mathbb{E}[\log p(x,y)]$.

Conditional: $H(Y|X) = H(X,Y) - H(X)$.

Mutual information: $I(X;Y) = H(X) + H(Y) - H(X,Y) = H(X) - H(X|Y) = \text{KL}(p(x,y) \| p(x)p(y)) \geq 0$.

Cross-entropy: $H(p,q) = -\mathbb{E}_p[\log q] = H(p) + \text{KL}(p\|q)$.

KL divergence: $\text{KL}(p\|q) = \mathbb{E}_p[\log(p/q)] \geq 0$, not symmetric.

12.2 Applications in ML/GNNs

Variational bound: ELBO in VAEs uses $\text{KL}(q\|p)$.

InfoMax in GNNs: Maximize $I(\mathbf{h}_v; \mathbf{h}_G)$ for unsupervised learning (e.g., InfoGraph).

Entropy regularization: In RL/policy, add $-H(\pi)$ to encourage exploration.

Chain rule: $H(X_1, \ldots, X_n) = \sum H(X_i | X_{<i})$.

Tricks: Jensen-Shannon divergence for stability: $\text{JSD}(p\|q) = \frac{1}{2}\text{KL}(p\|m) + \frac{1}{2}\text{KL}(q\|m)$, $m = (p+q)/2$.

In GNNs: Use MI to measure information flow between layers or nodes.

13 Anomaly Detection

13.1 Key Concepts & Definitions

Anomaly (outlier): Data point deviating significantly from normal patterns. Types:

- **Unsupervised**: No labels; e.g., assume most data normal.
- **Semi-supervised**: Train on normal data only (one-class).
- **Supervised**: Labeled anomalies (rare due to imbalance).

Challenges: High dims (curse of dimensionality), imbalance, thresholding.

Tricks: Normalize data; use dimensionality reduction (e.g., PCA) pre-detection; evaluate with AUC-PR over AUC-ROC for imbalance.

13.2 Statistical Methods

Z-Score: Score $z_i = \frac{x_i - \mu}{\sigma}$. Anomaly if $|z_i| > \theta$ (e.g., 3).

Mahalanobis Distance: Accounts for covariance.

$$D_M(\mathbf{x}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})}$$

Anomaly if $D_M > \theta$ (e.g., from χ^2 dist.).

13.3 Proximity-Based Methods

k-NN Outlier Score: Distance to k-th nearest neighbor $d_k(\mathbf{x})$. Anomaly if $d_k > \theta$.

Local Outlier Factor (LOF): Compares local density.

1. Reachability dist.: $\text{rd}_k(p,o) = \max(d_k(o), d(p,o))$.
2. Local reach. density: $\text{lrd}_k(p) = \left(\frac{1}{N_k(p)}\sum_{o \in N_k(p)} \text{rd}_k(p,o)\right)^{-1}$.
3. LOF: $\text{LOF}_k(p) = \frac{1}{N_k(p)}\sum_{o \in N_k(p)} \frac{\text{lrd}_k(o)}{\text{lrd}_k(p)}$.

Anomaly if $\text{LOF} > 1$ (much lower local density).

13.4 Isolation Forest

Ensemble of isolation trees: Randomly partition until isolation.

Anomaly Score: $s(\mathbf{x}, n) = 2^{-\frac{E(h(\mathbf{x}))}{c(n)}}$, where $h(\mathbf{x})$ = path length, $E(\cdot)$ = avg over trees, $c(n) = 2H(n-1) - \frac{2(n-1)}{n}$ (H = harmonic number). Anomaly if $s \approx 0.5$ (normal) or $s \to 1$ (anomaly). Trick: Works well in high dims; depth limit for efficiency.

13.5 One-Class SVM

Hyperplane maximizing margin from origin (normal data).

$$\min_{\mathbf{w}, \xi_i, \rho} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{\nu n}\sum_i \xi_i - \rho$$

s.t. $\mathbf{w}^T\phi(\mathbf{x}_i) \geq \rho - \xi_i$. Decision: $f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\phi(\mathbf{x}) - \rho)$. $\nu \in (0, 1]$ bounds outlier fraction.

13.6 Autoencoder-Based

Train on normal data; anomaly score = reconstruction error $||\mathbf{x} - \hat{\mathbf{x}}||^2$. Threshold via validation set.

13.7 Evaluation

Anomaly Score Thresholding: Use quantiles or ROC curve. Metrics: Precision@K, AUC-PR (for imbalance).

14 RL & Active Learning

14.1 Markov Decision Processes (MDPs)

MDP: Tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma, \mu_0)$, where $\mathcal{S}$ states, $\mathcal{A}$ actions, $P(s'|s, a)$ transition prob., $R(s, a)$ reward, $\gamma \in [0, 1)$ discount, μ_0 initial state dist.

State-Value Fn: $V^\pi(s) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s\right]$.

Action-Value Fn: $Q^\pi(s, a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s, a_0 = a\right]$.

Advantage Fn: $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$.

Bellman Expectation: $V^\pi(s) = \sum_a \pi(a|s)\sum_{s',r} P(s', r|s, a)[r + \gamma V^\pi(s')]$.

Bellman Optimality: $V^*(s) = \max_a \sum_{s',r} P(s', r|s, a)[r + \gamma V^*(s')]$. Trick: Optimal policy $\pi^*(s) = \text{argmax}_a Q^*(s, a)$.

Discounted Return: $G_t = \sum_{k=t}^{\infty} \gamma^{k-t} R_{k+1}$. Exam hint: Use for infinite-horizon problems; $\gamma < 1$ ensures convergence.

14.2 Value & Policy Iteration

Value Iteration (VI): Update $V(s) \leftarrow \max_a \mathbb{E}[R(s, a) + \gamma V(s')]$. Converges to V^* (contraction mapping, Banach fixed-point thm). Trick: Stop when $\max_s |V_{new}(s) - V_{old}(s)| < \epsilon(1-\gamma)/\gamma$.

Policy Iteration (PI): (1) Eval: Solve $V^\pi = (I - \gamma P^\pi)^{-1}R^\pi$. (2) Improve: $\pi'(s) = \text{argmax}_a Q^\pi(s, a)$. Faster than VI for small $\mathcal{A}$.

Exam trick: PI is exact eval + greedy; VI is approx. eval + greedy. Derive from Bellman.

14.3 Model-Free RL (TD Methods)

TD(0) Update: $V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)]$ (TD error: $r + \gamma V(s') - V(s)$).

Q-Learning (Off-Policy): $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$. ϵ-greedy exploration.

SARSA (On-Policy): $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$.

Trick: Q-Learning converges to optimal even with suboptimal policy; SARSA to policy's Q. Exam: Compare bias/variance.

14.4 Policy Gradients & Actor-Critic

Policy Gradient Thm: $\nabla_\theta J(\theta) = \mathbb{E}_\pi[\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s, a)]$. Update: $\theta \leftarrow \theta + \alpha\nabla_\theta J$.

REINFORCE: Monte-Carlo est. $\hat{\nabla} J = \sum_t \nabla_\theta \log \pi(a_t|s_t) G_t$. Variance reduction: Subtract baseline $b(s_t) \approx V(s_t)$.

Actor-Critic: Actor updates policy; Critic est. V or Q via TD. A2C/A3C: Advantage $A = r + \gamma V(s') - V(s)$.

Trick: Softmax policy $\pi(a|s) = \exp(h_\theta(s, a))/\sum_{a'}\exp(h_\theta(s, a'))$; gradient $\nabla \log \pi = \psi(s, a) - \sum_{a'} \pi(a'|s)\psi(s, a')$ (compat. features ψ).

Exam calc: Derive policy grad from $\frac{\partial}{\partial \theta}\mathbb{E}[R] = \mathbb{E}[R \nabla \log p(R|\theta)]$ (score fn trick).

14.5 Active Learning

Pool-based: Unlabeled pool $\mathcal{U}$, labeled $\mathcal{L}$. Query strategy selects $x^* \in \mathcal{U}$ to label.

Uncertainty Sampling: Query $x^* = \text{argmax}_x H(y|x, \mathcal{L})$ or $\text{argmax}_x[1 - P(\hat{y}|x)]$ (least confident). For binary: $\text{argmax}_x \min(P(y = 1|x), P(y = 0|x))$.

Query-by-Committee: Train committee of models; query max disagreement (vote entropy: $H = -\sum_y V(y)/C \log V(y)/C$, $V(y)$ votes for y).

Expected Model Change: Query $\text{argmax}_x \mathbb{E}_{y|x}[||\nabla \ell(y, f(x))||]$.

Trick: Balances exploration (uncertainty) vs. exploitation. Exam: Compare to random sampling; derive entropy for multi-class.

Bayesian Active Learning: Use GP or BNN for $p(y|x)$; query max info gain $I(y; x) = H(y) - \mathbb{E}_{p(x)}[H(y|x)]$.

Exam hint: Active learning reduces labeling cost; focus on info-theoretic justifications.

15 Counterfactual Invariance

Key: Models robust to interventions (do-operator). Exam hint: Check invariance via causal graphs (SCMs); derive counterfactuals from joint distributions.

Definition 1 (Structural Causal Model (SCM))**.** SCM: $X_i = f_i(\mathbf{PA}_i, U_i)$, where $\mathbf{PA}_i$ are parents, U_i noise. Intervention $\text{do}(X = x)$: Replace f_X with constant x.

Formulas:

$$P(Y|do(X = x)) = \sum_z P(Y|X = x, Z = z)P(Z|X = x) \quad ($$

$$\text{Counterfactual} : P(Y_x = y|Y_{x'} = y') \quad (\text{"What if X was x given}$$

Tricks/Hints:

- Invariance: Model f is counterfactually invariant if $f(X, do(A)) = f(X)$ for action A (e.g., distribution shift robustness).
- Exam proof: Use Pearl's ladder (observational $\to$ interventional $\to$ counterfactual). Check d-separation for identifiability.

Key Invariance Condition

Invariant if $\mathbb{E}[Y|X, E] = \mathbb{E}[Y|X]$ for environment E (no confounding).

16 Reproducing Kernel Hilbert Spaces (RKHS)

Key: Space $\mathcal{H}$ of functions with kernel K. Exam hint: Prove properties via inner products; compute norms for regularization.

Definition 2 (RKHS)**.** Hilbert space $\mathcal{H}$ where eval $f \mapsto f(x)$ continuous. Reproducing: $f(x) = \langle f, K(\cdot, x)\rangle_\mathcal{H}$.

Formulas:

$$K(x, y) = \langle \phi(x), \phi(y)\rangle_\mathcal{H} \quad \text{(Kernel trick, feature map } \phi)$$
$$\langle K(\cdot, x), K(\cdot, y)\rangle_\mathcal{H} = K(x, y) \quad \text{(Reproducing property)}$$
$$\|f\|_\mathcal{H}^2 = \sum_{i,j} \alpha_i\alpha_j K(x_i, x_j) \quad \text{(For } f(\cdot) = \sum_i \alpha_i K(\cdot, x_i))$$

$$\text{Mercer's Thm: } K(x, y) = \sum_{k=1}^{\infty} \lambda_k e_k(x) e_k(y) \quad \text{(Pos. def. kernel}$$

Tricks/Hints:

- Positive definite: K pos. def. if $\sum_{i,j} c_i c_j K(x_i, x_j) \geq 0 \, \forall \mathbf{c} \neq 0$.
- Kernel ridge reg.: $\hat{f} = \text{argmin}_f \|f\|^2_{\mathcal{H}} + \frac{1}{n}\sum_i (y_i - f(x_i))^2$. Sol: $\boldsymbol{\alpha} = (\mathbf{K} + \lambda \mathbf{I})^{-1}\mathbf{y}$.
- Exam: For new x_*, predict $f(x_*) = \mathbf{k}_*^T \boldsymbol{\alpha}$, where $k_{*i} = K(x_*, x_i)$.

Common Kernels

Linear:	$K(x,y) = x^T y$
RBF:	$K(x,y) = \exp(-\|x-y\|^2/2\sigma^2)$
Polynomial:	$K(x,y) = (x^T y + c)^d$

Methods

VAEs & Non-Parametric Bayesian

17 Variational Autoencoders (VAEs)

17.1 Key Concepts & Setup

VAE: Generative model with latent $z \sim \mathcal{N}(0, I)$. Encoder $q_\phi(z|x)$ (neural net) approximates posterior $p(z|x)$. Decoder $p_\theta(x|z)$ reconstructs x. Goal: Maximize marginal log-likelihood $\log p(x) = \mathbb{E}_{q(z|x)}[\log p(x|z)] - \text{KL}(q(z|x)\|p(z|x))$.

Trick: Use ELBO as surrogate (variational lower bound).

17.2 ELBO Formula

$$\text{ELBO}(\theta, \phi) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \text{KL}(q_\phi(z|x)\|p(z))$$

- Reconstruction term: Measures data fit (e.g., Bernoulli or Gaussian likelihood). - KL term: Regularizes encoder to match prior (closed-form for Gaussians: $\text{KL}(\mathcal{N}(\mu,\sigma^2)\|\mathcal{N}(0,1)) = \frac{1}{2}\sum(1 + \log\sigma^2 - \mu^2 - \sigma^2)$). - Exam hint: Derive by Jensen's inequality; $\log p(x) \geq \text{ELBO}$.

17.3 Reparameterization Trick

For backprop through sampling: $z = \mu + \sigma \odot \epsilon, \epsilon \sim \mathcal{N}(0, I)$. Allows gradient $\nabla_\phi \mathbb{E}_{q_\phi(z|x)}[f(z)] \approx \nabla_\phi f(\mu + \sigma \odot \epsilon)$.

Trick: Use for stochastic optimization; avoids high-variance Monte Carlo.

17.4 Training & Hints

- Loss: $-$ELBO (minimize via SGD/Adam). - β-VAE: Weight KL term by $\beta > 1$ for disentangled latents. - Exam-relevant: VAEs vs. GANs (VAEs stable, probabilistic); limitations (blurry outputs due to pixel-wise loss).

18 Non-Parametric Bayesian Methods

18.1 Gaussian Processes (GPs)

GP: $f(x) \sim \text{GP}(m(x), k(x, x'))$, mean $m(\cdot)$ (often 0), kernel $k(\cdot,\cdot)$ (e.g., RBF: $k(x,x') = \exp(-\|x - x'\|^2/2\ell^2)$).

Predictive Distribution (Regression, noisy $y = f(x) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma^2)$): Let X_*, y_* for test, X, y for train. Kernel matrix $K = k(X,X) + \sigma^2 I$.

$$\mu_* = k(X_*, X)(K)^{-1}y, \quad \Sigma_* = k(X_*, X_*) - k(X_*, X)(K)^{-1}k(X, X_*)$$

- Posterior: $p(f_*|X_*, X, y) = \mathcal{N}(\mu_*, \Sigma_*)$. - Log-marginal likelihood: $\log p(y|X) = -\frac{1}{2}y^T K^{-1} y - \frac{1}{2}\log|K| - \frac{n}{2}\log 2\pi$ (for hyperparam tuning).

18.2 Kernels & Tricks

- Valid kernels: Positive semi-definite (e.g., linear, polynomial, Matérn). - Trick: Kernel trick for non-linear regression without explicit features. - Composition: Sum/product of kernels for flexibility. - Exam hint: Compute K matrix for small datasets; derive posterior mean/variance.

18.3 Dirichlet Processes (DPs) & Infinite Mixtures

DP: $\text{DP}(\alpha, H)$ for infinite mixture models (e.g., Dirichlet Process Mixture Model for clustering). Stick-breaking: $G = \sum_{k=1}^\infty \pi_k \delta_{\theta_k}$, $\pi_k = v_k \text{prod}_{j=1}^{k-1}(1 - v_j)$, $v_j \sim \beta(1, \alpha)$. - Exam-relevant: Non-parametric alternative to finite GMMs; allows model complexity to grow with data. - Hint: Chinese Restaurant Process analogy for sampling.

18.4 Hints for Non-Parametrics

- GPs vs. NNs: GPs exact uncertainty, but $O(n^3)$ cost (use sparse approximations). - Use for Bayesian optimization or regression with small data.

19 PAC Learning

19.1 Key Definitions & Framework

- **Hypothesis class** $\mathcal{H}$: Set of functions $h : \mathcal{X} \to \mathcal{Y}$ (e.g., binary classifiers, $\mathcal{Y} = \{0, 1\}$).
- **Realizable PAC**: $\exists h^* \in \mathcal{H}$ with true risk $L(h^*) = 0$.
- **PAC Learnable**: $\exists$ learner s.t. $\forall$ distributions $\mathcal{D}, \forall \epsilon, \delta > 0$, with prob. $\geq 1 - \delta$, outputs h with $L(h) \leq \epsilon$ using $m = m(\epsilon, \delta)$ samples.
- **Agnostic PAC**: No assumption on h^*; minimize excess risk over $\mathcal{H}$.
- **True/Generalization Risk**: $L(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(h(x), y)]$ (e.g., 0-1 loss: $\ell = \mathbf{1}_{h(x)\neq y}$).
- **Empirical Risk**: $\hat{L}(h) = \frac{1}{m}\sum_{i=1}^m \ell(h(x_i), y_i)$.

19.2 VC Dimension & Shattering

- **Shattering**: $\mathcal{H}$ shatters set $S \subseteq \mathcal{X}$ if $|\{\mathbf{y} \in \{0,1\}^{|S|} : \exists h \in \mathcal{H} \text{ realizes } \mathbf{y} \text{ on } S\}| = 2^{|S|}$.
- **Growth Function**: $\Pi_{\mathcal{H}}(m) = \max_{S:|S|=m} |\{h|_S : h \in \mathcal{H}\}| \leq \left(\frac{em}{d}\right)^d$ (Sauer-Shelah, if VC-dim $d < \infty$).
- **VC Dimension** $d = \text{VC}(\mathcal{H})$: Largest $|S|$ s.t. $\mathcal{H}$ shatters S (infinite if no such max).
- **Examples**: VC(intervals on $\mathbb{R}$) = 2; VC(linear classifiers in $\mathbb{R}^d$) $= d + 1$; VC(axis-aligned rectangles in $\mathbb{R}^2$) = 4.
- **Trick for VC Calc**: Find largest shatterable set (e.g., for half-planes: 3 points not collinear shatter, 4 do not).

19.3 Key Formulas & Bounds

Fundamental Thm of PAC (Realizable, Finite $\mathcal{H}$): $m \geq \frac{1}{\epsilon}(\ln|\mathcal{H}| + \ln(1/\delta))$ samples suffice for $L(h) \leq \epsilon$ w.p. $\geq 1 - \delta$ via ERM.

Infinite $\mathcal{H}$ (VC-based): For VC-dim d, $m \geq C\frac{d + \ln(1/\delta)}{\epsilon}$ (lower bound); upper: $m = O\left(\frac{d\ln(1/\epsilon) + \ln(1/\delta)}{\epsilon}\right)$.

Agnostic PAC (Uniform Convergence): w.p. $\geq 1 - \delta$,

$$|L(h) - \hat{L}(h)| \leq \sqrt{\frac{2d\ln(em/d) + \ln(2/\delta)}{m}} \quad \text{(Rademacher/VC}$$

Sample Complexity (Agnostic): $m = O\left(\frac{d\ln(1/\epsilon) + \ln(1/\delta)}{\epsilon^2}\right)$ for excess risk $\leq \epsilon$.

Tricks:

- Use Hoeffding for finite $|\mathcal{H}|$: $\Pr(|L - \hat{L}| > \epsilon) \leq 2|\mathcal{H}|e^{-2m\epsilon^2}$.
- For VC, bound $\Pi_{\mathcal{H}}(m) \leq \sum_{i=0}^d \binom{m}{i} \leq (em/d)^d$.
- ERM is PAC if $\mathcal{H}$ has finite VC-dim.