EASYDEEPLEARN

Unsupervised Learning

Interview cheat sheet

214 questions · Clustering, dimensionality reduction, density estimation and anomaly detection.

#clustering #applications #representation-learning #dimensionality-reduction #interview #anomaly-detection #nlp #density #evaluation #deep-learning

Clustering basics

How does k-means work and what are its main limitations?

Initialize k centroids (k-means++ is best). Assign each point to the nearest centroid, then recompute centroids as the mean of assigned points; repeat until convergence.

How do you choose the number of clusters k?

Use multiple signals, not one: the elbow method on inertia (look for the bend), the silhouette score (higher is better, up to 1), the gap statistic, and domain knowledge. For density-based clustering (DBSCAN, HDBSCAN) you don't set k at all — you set density parameters instead.

When would you pick DBSCAN over k-means?

Pick DBSCAN when clusters are non-convex or of very different densities, when you don't want to specify k, and when you want an explicit notion of noise/outliers. k-means struggles with irregular shapes and forces every point into a cluster. DBSCAN needs two parameters: eps (neighborhood radius) and $\min_{\text{samples}}$.

What does the silhouette score measure?

For each point, silhouette = (b - a) / max(a, b), where a is the mean distance to points in its own cluster and b is the mean distance to points in the nearest other cluster. Values close to +1 mean well-separated clusters, near 0 means overlapping, negative means misassigned.

What is hierarchical clustering and when is it useful?

It builds a tree (dendrogram) of nested clusters, either agglomerative (bottom-up: merge closest clusters) or divisive (top-down). You cut the tree at a chosen height to get k clusters.

Gaussian Mixture Model vs k-means — what's the difference?

k-means gives hard assignments and assumes spherical, equal-variance clusters. A GMM models each cluster as a multivariate Gaussian and returns soft (probabilistic) memberships; it handles ellipsoidal clusters and different covariances.

What is linkage in agglomerative clustering?

Linkage defines the distance between two clusters. Single linkage = min pairwise distance (chains, elongated clusters).

Why does clustering degrade in high dimensions?

In high dimensions, pairwise distances concentrate — points look almost equidistant — so distance-based clustering loses discriminative power. Also, irrelevant features add noise.

How does k-means++ initialization work?

Pick the first centroid uniformly at random. For each subsequent centroid, sample point x with probability proportional to $D(x)^2$ where D(x) is the distance to the nearest already-chosen centroid.

When is mini-batch k-means preferred?

Large datasets (millions of points). Sample small random mini-batches (~1000 points), update centroids incrementally toward batch means with a learning rate.

k-medoids vs k-means — the difference.

k-medoids uses actual data points as cluster representatives (medoids) instead of arithmetic means. Robust to outliers (the median is more robust than the mean).

What is fuzzy c-means?

Each point has a membership degree $u_{ij} \in [0,1]$ for every cluster (rows sum to 1). Centroids weighted by u_{ij}^m for a fuzzifier m > 1 (m=2 typical).

HDBSCAN — how does it improve on DBSCAN?

Runs DBSCAN over all density levels and extracts the most stable clusters — avoids picking a single eps. Handles clusters of varying density (DBSCAN's biggest failure mode).

OPTICS — what does it produce and how do you use it?

Computes a reachability plot ordering all points by density-reachability. Reveals cluster hierarchy visually as valleys in the plot.

Spectral clustering — the core idea.

Build similarity graph on data (k-NN or ε-neighborhood). Compute graph Laplacian L.

Affinity propagation — how does it work?

Message-passing algorithm exchanging 'responsibility' and 'availability' messages between all pairs of points until exemplars emerge. No k needed — 'preference' parameter influences the number of clusters.

Mean shift clustering — mechanism and use case.

Non-parametric mode-seeking. For each point, repeatedly shift it toward the local mean within a bandwidth-radius kernel until convergence.

BIRCH — when to use it?

Streaming / very large dataset clustering. Builds a CF-Tree (Cluster Feature summaries: N, LS, SS) in a single pass over data — leaves become micro-clusters.

EM for a GMM — one iteration explicitly.

E-step: for each point x_i and cluster j, $\gamma_{ij} = \pi_j \, N(x_i \mid \mu_j, \Sigma_j) \,/\, \Sigma_k \, \pi_k \, N(x_i \mid \mu_k, \Sigma_k)$ (responsibilities). M-step: update $\pi_j = \Sigma_i \, \gamma_{ij} \,/\, n$, $\mu_j = \Sigma_i \, \gamma_{ij} \, x_i \,/\, \Sigma_i \, \gamma_{ij}$, $\Sigma_j = \Sigma_i \, \gamma_{ij} \, (x_i - \mu_j)(x_i - \mu_j)' \,/\, \Sigma_i \, \gamma_{ij}$.

GMM covariance types — which do you pick?

Full: each cluster has its own general Σ_j — most flexible, most parameters (p^2). Tied: all clusters share one Σ.

How does BIC choose k in GMM?

BIC = -2 log L + p log n where p = number of free parameters, n = dataset size. Lower is better.

How does the gap statistic work?

Compare observed within-cluster dispersion W_k to expected W_k under a null reference distribution (uniform in the bounding box). $\mathrm{Gap}(k) = E \cdot [\log W_{k\cdot}] - \log W_k$.

Why can the elbow method fail?

(1) No clear elbow — smooth curve, subjective. (2) Different features scales dominate SSE → misleading k.

Calinski-Harabasz index — what is it?

$\mathrm{CH}(k) = [\mathrm{Tr}(B_k) / (k-1)] / [\mathrm{Tr}(W_k) / (n-k)]$ — ratio of between-cluster dispersion to within-cluster dispersion (higher = better). Analogue to an F-statistic for clustering.

Davies-Bouldin index — how is it computed?

DB = (1/k) $\Sigma_i \max_{j \neq i} [(s_i + s_j) / d_{\mathrm{ij}}]$ where $s_i = \mathrm{within-cluster}$ scatter, $d_{\mathrm{ij}} = \mathrm{distance}$ between centroids. Lower is better.

Adjusted Rand Index vs NMI — external cluster metrics.

Both compare a clustering to ground-truth labels (when available). ARI = (RI - E[RI]) / (max RI - E[RI]) → 0 for random, 1 for perfect match, can be negative.

How do you cluster mixed numeric + categorical data?

(1) Gower distance — averages appropriate distances per feature type (Manhattan for numeric, matching for categorical). (2) k-prototypes — extends k-means with categorical dissimilarity.

Why do you scale features before k-means?

Euclidean distance is scale-sensitive: a feature with 1000-range dominates one with 0-1 range. Standardize (z-score) so features contribute equally.

Cosine similarity vs Euclidean — when do you use cosine?

Cosine similarity = x · y / (|| x || || y ||) — measures angle, ignores magnitude. Use when magnitude is irrelevant: TF-IDF vectors (doc length varies), sentence embeddings, user-item preferences.

How do outliers affect k-means and how do you handle them?

Outliers pull centroids toward themselves (mean is non-robust) → distort cluster boundaries. Handle: (1) pre-cluster outlier removal via IsolationForest / z-score, (2) use k-medoids (median-like), (3) use DBSCAN which explicitly flags outliers as noise, (4) trim tails / winsorize before clustering.

How do you assess cluster stability?

(1) Bootstrap: resample data, re-cluster, measure Jaccard overlap between cluster memberships. (2) Random-init variability: many k-means restarts, measure ARI between runs.

What is consensus clustering?

Run clustering many times (varying algorithm, k, subsample, init). Build co-clustering matrix $M_{\mathrm{ij}} = \mathrm{fraction}$ of runs where i and j were in the same cluster.

Online k-means — how does it work?

For each incoming point, find nearest centroid and update it: $\mu_j \leftarrow \mu_j + \eta\,(x - \mu_j)$ with decaying learning rate. Streaming-friendly, constant memory.

'Cluster then classify' — when does it help?

Add cluster label as an extra feature to a supervised model. Sometimes lifts performance on data with strong latent groups (customer segments, region clusters) — the label encodes a non-linear grouping the model wouldn't easily learn.

How do you cluster time series?

(1) Feature extraction: extract statistics (mean, variance, autocorrelation, FFT bands) and cluster the feature vectors. (2) Dynamic Time Warping (DTW) distance + k-medoids / hierarchical.

What is Dynamic Time Warping?

Similarity between two sequences allowing non-linear alignment (stretch/compress). Finds the best matching via dynamic programming through a cost matrix.

Community detection in graphs — main approaches.

(1) Modularity optimization: Louvain, Leiden (greedy hierarchical) — scale to millions of nodes, standard default. (2) Spectral: eigenvectors of the graph Laplacian → k-means.

How do you interpret / visualize clusters?

(1) 2D projection with UMAP / t-SNE colored by cluster. (2) Feature-importance-per-cluster: mean of each feature per cluster vs global mean → describes each group.

How do you find nested or hierarchical structure?

(1) Agglomerative clustering with a dendrogram → cut at different heights for different granularity. (2) HDBSCAN's condensed cluster tree — natural hierarchy.

Dirichlet Process — how does it help clustering?

Bayesian nonparametric: number of clusters is inferred rather than pre-specified. New points can start new clusters with a probability depending on concentration parameter α (Chinese Restaurant Process metaphor: 'rich get richer' + occasional new table).

One cluster dominates in k-means — what do you do?

(1) Try higher k — the 'big' cluster may naturally split. (2) Log-scale skewed features.

Interview: 'you're asked to segment 10M customers — how?'

(1) Sample down to 1M for prototype. (2) Feature engineering: RFM (recency, frequency, monetary), behavioral counts, categorical embeddings.

How do you monitor cluster drift over time?

Track: (1) fraction of new points assigned to each cluster (should be stable). (2) Distribution of intra-cluster distances (blow-up = drift).

Mixture models for density estimation — beyond GMM.

GMM assumes Gaussian components. Alternatives: mixture of t-distributions (heavier tails, robust), mixture of Dirichlet (categorical), mixture density networks (NN outputs mixture parameters — good for heteroscedastic regression), Bayesian nonparametric DP-mixtures (infer k).

LDA vs NMF for topic modeling — which do you pick?

LDA: probabilistic — documents = Dirichlet mixture of topics, topics = Dirichlet mixture over words. Fit via variational Bayes or Gibbs.

How does BERTopic work?

(1) Embed documents with sentence-transformers. (2) Reduce dimensionality with UMAP.

Applications of clustering in NLP.

(1) Topic modeling / document grouping (LDA, NMF, BERTopic). (2) Word sense discovery: cluster contextualized embeddings of ambiguous words.

How do you cluster images at scale?

(1) Feature extraction: pass through pretrained CNN or ViT (DINOv2, CLIP) → embeddings. (2) UMAP / PCA to ~50 dim for compute.

User behavior segmentation — feature engineering.

(1) RFM: Recency (days since last activity), Frequency (activities per period), Monetary (spend). (2) Behavioral counts: events per day/week.

Clustering in single-cell RNA-seq — the standard pipeline.

(1) QC + normalize (log-CPM, SCTransform). (2) HVG selection.

Interview: cluster 5M e-commerce customers for a marketing campaign.

(1) Define business goal (targeting, retention, cross-sell?). (2) Sample 500k for prototyping.

Interview: how would you group log messages from a large distributed system?

(1) Parse log: extract template (Drain algorithm, LogPunk) → replace variables (timestamps, IPs, numbers) with placeholders. (2) Vectorize templates via TF-IDF or sentence-transformers on remaining text.

Interview: business asks 'why is customer X in Cluster 3?' — how do you explain?

(1) Show customer X's features + Cluster 3 mean/std for each feature → highlight top-3 features driving assignment (largest z-score contribution). (2) Show representative medoid / centroid of Cluster 3 alongside X.

Interview: 'you have to pick k for the executive team — walk me through it.'

(1) Business first: how many actionable groups can marketing / product operate? Usually 3-8 in practice.

Interview: after segmentation, one segment has 60% women but each region shows 40%. Why?

Simpson's paradox — different regions have different sizes / proportions in that cluster. Verify with contingency table: within each region, the segment may be 40% women, but the sample-size-weighted overall proportion is 60% because certain regions dominate the segment.

Interview: how would you cluster 1B rows on a budget?

(1) Sample 1M for prototype clustering. (2) Feature-engineer + reduce (PCA / hashed features).

Interview: you cluster and 3 clusters look 'right', but 2 mix categories — what next?

(1) Feature analysis: what's shared in each mixed cluster (maybe a real behavior spans your labels). (2) Try higher k → often 'mixed' clusters split cleanly.

Interview: 'no labels — how do you know your clustering is any good?'

(1) Internal indices: silhouette + Calinski-Harabasz + Davies-Bouldin across k. (2) Stability: bootstrap Jaccard, ARI between init runs.

Interview: 'when should you NOT cluster?'

(1) Data is truly unimodal / uniform — no structure to find. (2) Very high dimensionality without reduction → distance concentration ruins similarity.

Interview: 'how would you tune HDBSCAN's $\min_{\text{cluster}}$ size?'

(1) Business constraint: what's the smallest meaningful group in the domain? (2) Sensitivity analysis: sweep $\min_{\text{cluster}}$ size ∈ [5, 10, 20, 50, 100] → measure silhouette + noise-fraction + number-of-clusters.

Interview: 'discuss the tradeoffs between k-means, DBSCAN, and HDBSCAN.'

(1) k-means: fast, scales to millions (mini-batch), but assumes spherical + equal-size clusters + fixed k + non-robust to outliers. (2) DBSCAN: handles non-convex + detects noise, but needs eps tuning + fails with varying densities + O(n log n) with index.

Interview: 'what's the biggest mistake you've seen in a real clustering project?'

(Answering as a candidate) '(1) Not scaling features — one feature dominated distances → nonsense clusters. (2) Trusting elbow / silhouette alone without stability check → non-reproducible clusters.

Without labels, how do you convince a stakeholder your clustering is any good?

Combine three kinds of evidence, because no single one is sufficient. Internal measures such as silhouette or Davies-Bouldin say whether clusters are compact and separated, which is necessary but not meaningful on its own.

You run k-means on customer data with age, income and number of purchases. What breaks?

The scales. Income in dollars ranges over tens of thousands while age spans a few decades, so squared Euclidean distance is essentially income distance and the other two variables contribute nothing.

The elbow plot has no elbow. How do you pick k?

Accept that a smooth curve is telling you the data has no crisp cluster count, which is common and not a failure of method. Look at the silhouette across a range of k and prefer a value that is locally best rather than globally optimal.

DBSCAN labels almost everything as noise. What do you change?

The neighbourhood radius is too small for the density of your data, or the minimum points is too large. Set the minimum points from domain reasoning, often around twice the dimensionality, then choose the radius from the k-distance plot: sort every point's distance to its k-th nearest neighbour and pick the value at the knee, which is where density drops off.

Is it a good idea to run k-means on raw text embeddings?

It works, with caveats worth stating. Embeddings are high-dimensional and roughly isotropic, so Euclidean distances concentrate and everything looks similarly far apart, which weakens the contrast k-means depends on.

LDA or embedding-based topic modelling for a corpus of support tickets?

Embedding-based clustering is usually the better modern default. LDA treats a document as a bag of words with no notion of synonymy, which hurts badly on short texts like tickets, where there are too few words per document to estimate a topic mixture reliably.

Your customer segments change completely when you re-run the pipeline monthly. Is that acceptable?

No, because a segmentation the business acts on has to be stable enough to plan around, and instability means you are describing sampling noise rather than structure. First check whether the instability is algorithmic: k-means with random initialization on data with no strong separation gives different partitions every run, which a fixed seed hides without fixing.

Advanced clustering & density

Local Outlier Factor (LOF) — how does it work?

For each point, compare its local density (via k-NN distances) to that of its neighbors. LOF > 1: point is in a lower-density region than its neighbors → outlier.

Elliptic envelope / robust Mahalanobis distance — when to use?

Fit a robust multivariate Gaussian (Minimum Covariance Determinant, Rousseeuw-Van Driessen) → flag points with high Mahalanobis distance. Assumes elliptical / Gaussian normal cloud; robust to outliers via MCD.

Kernel Density Estimation — mechanism.

Non-parametric density estimator: $\hat{p}(x) = (1/(nh^d)) \Sigma K((x - x_i)/h)$. K = kernel (Gaussian typical). h = bandwidth (critical hyperparameter).

How do you choose the KDE bandwidth?

(1) Silverman's rule: h = (4/(d+2))^(1/(d+4)) * σ * n^(-1/(d+4)) — closed form, assumes Gaussian data. (2) Scott's rule: similar but different exponent.

What is a copula and why use one?

Copula separates marginal distributions from dependence structure: $F(x, y) = C(F_X(x), F_Y(y))$. Sklar's theorem: unique C for continuous margins.

Normalizing flows for density estimation — the idea.

Learn invertible neural network f_θ that maps simple base distribution (Gaussian) to complex data distribution. Change of variables: log p(x) = log $p(z) + \log |\det J_f$^(-1)|.

Score matching — what does it estimate?

Rather than p(x), estimate the score s(x) = ∇_x log p(x). Loss (Hyvärinen): $E[|| s\theta(x) - \nabla \log p(x) ||^2]$ → doesn't need normalization constant → tractable for unnormalized density models.

KL divergence — what it measures and pitfalls.

$KL(P || Q) = \Sigma$ P(x) $\log(P(x)/Q(x))$ = -H(P) + H(P, Q). Not symmetric, not a metric.

Distance & scaling considerations

HNSW — how does it work?

Hierarchical multi-layer graph. Top layer: sparse long-range edges.

Product Quantization (PQ) — how does it compress vectors?

Split d-dim vector into m sub-vectors of d/m dims. Cluster each sub-vector space separately (k centroids per subspace, k typically 256).

Locality-Sensitive Hashing (LSH) — the core trick.

Hash function h such that P(h(x) = h(y)) is high for close x, y and low for far ones (opposite of crypto hash). Different metric → different LSH family: MinHash for Jaccard (near-duplicate documents), SimHash for cosine, random hyperplane for cosine, Euclidean via random projections.

MinHash — how does it estimate Jaccard similarity?

Given sets A, B: pick many random permutations of the universe. For each permutation, MinHash(A) = smallest element of A under that permutation.

Interview: your image dataset (10M) has near-duplicates — how do you dedup at scale?

(1) Perceptual hashing (pHash / dHash / wHash): 64-bit hash → Hamming distance threshold catches near-duplicates and small edits. Fast, cheap.

Concretely, what goes wrong with distance-based methods in high dimensions?

Distances concentrate: as dimensionality grows, the ratio between the nearest and farthest neighbour of a point tends towards one, so the very notion of a nearest neighbour loses discriminative power. Volume also grows so fast that any realistic sample is sparse, meaning every point is effectively an outlier and density estimates have almost no support.

Cluster evaluation

How do you evaluate anomaly detection?

Usually severely imbalanced → don't use accuracy. Standard metrics: (1) ROC AUC (rank-based, threshold-free), (2) PR AUC (better under extreme imbalance), (3) precision@k and recall@k at operating threshold, (4) F1 for a chosen threshold.

How do you evaluate the quality of a self-supervised representation?

(1) Linear probing: freeze encoder, train linear classifier on downstream labels → measures 'linearly separable info'. (2) Fine-tuning: unfreeze, full downstream training → measures 'total info'.

How do you evaluate topic quality?

(1) Coherence metrics: C_V, C_{UMass}, C_{NPMI} (Röder et al.) — measure semantic similarity of top-k topic words. (2) Topic diversity: fraction of unique top-k words across topics.

Your anomaly detector has no labels. How do you set the decision threshold?

Set it from operational capacity rather than from statistics. Decide how many alerts a human can genuinely review per day, then take that quantile of the score distribution, which turns an unanswerable question into a staffing one.

What makes a good self-supervised pretext task?

It must be solvable only by learning something you actually want. The failure mode is a shortcut: a task the network can solve with a superficial cue, such as detecting a rotation from a border artefact or matching two crops by their shared colour histogram, which yields a high pretext score and useless representations.

Linear dimensionality reduction

PCA vs t-SNE vs UMAP — when do you use each?

PCA is a linear, deterministic projection that preserves global variance — use it for compression, denoising, or as input to another model. t-SNE and UMAP are nonlinear and preserve local neighborhoods — use them for 2D/3D visualization, not for downstream modeling. UMAP is faster than t-SNE and preserves more global structure.

How do you decide how many PCA components to keep?

Look at the cumulative explained-variance ratio and keep enough components to reach a target (e.g., 90% or 95%). You can also inspect the scree plot for an elbow, or pick components based on downstream cross-validation performance.

Derive PCA — what does it optimize?

Given centered X, find orthonormal w that maximizes $\mathrm{Var}(\mathrm{Xw}) = w$' Σ w subject to ||w||=1. Solution: eigenvector of Σ = X'X/n corresponding to the largest eigenvalue.

PCA via SVD — the connection.

For centered X (n × p), SVD gives X = UΣV'. Principal directions = columns of V (right singular vectors).

Why standardize before PCA (usually)?

PCA maximizes variance in the original units. If features have wildly different scales (income in $ vs age in years), the high-variance one dominates → useless components.

When does PCA fail?

(1) Non-linear structure (rolls, spirals) — PCA sees only linear correlations. (2) Discrete / categorical features (one-hot creates artificial variance).

Kernel PCA — when and how?

Apply the kernel trick to PCA: implicitly map data to a feature space Φ(x) via kernel K(x, y) = ⟨Φ(x), Φ(y)⟩, then do PCA there. Captures non-linear structure without explicit feature construction.

ICA vs PCA — the key difference.

PCA: finds uncorrelated components maximizing variance. ICA: finds statistically independent components — stronger requirement.

Non-negative Matrix Factorization (NMF) — when to use?

Factor X ≈ WH with W, H ≥ 0. Enforces additive parts-based representation.

Sparse coding — what is it?

Learn overcomplete dictionary D such that x ≈ Dα with sparse α (few non-zero entries). Uses: image compression, denoising, feature learning.

Random projection — how does it work?

Johnson-Lindenstrauss: for ε > 0, a random Gaussian matrix R (m × p) with $m = O(\log\ n\ /\ \varepsilon^2)$ preserves pairwise distances up to (1±ε) with high probability. Extremely cheap (no data-dependent fit).

Linear Discriminant Analysis (LDA) vs PCA — the difference.

PCA is unsupervised — max variance. LDA is supervised — projects to maximize class separability: max $\mathrm{Tr}(S_B)\ /\ \mathrm{Tr}(S_W)$, where $S_B = \mathrm{between-class\ scatter}$, $S_W = \mathrm{within-class\ scatter}$.

Canonical Correlation Analysis (CCA) — use case.

Finds pairs of linear projections (u, v) of two multivariate datasets X, Y that maximize their correlation. Uses: (1) multi-view learning — combine images and text embeddings, (2) genomics (SNPs vs gene expression), (3) recommendation with side information.

t-SNE perplexity — what does it control?

Perplexity ≈ effective number of neighbors each point 'considers' when building the conditional similarity distribution. Typical range: 5-50.

Top t-SNE pitfalls to avoid.

(1) Cluster sizes in the map are not meaningful — t-SNE inflates dense clusters. (2) Inter-cluster distances are NOT preserved — 'far' clusters aren't necessarily far in data space.

UMAP vs t-SNE — practical differences.

UMAP: faster $(O(n^{1}14)$ vs $O(n\ \log\ n)$ $\mathrm{with\ Barnes-Hut\ but\ higher\ constant})$, preserves more global structure (inter-cluster distances less-meaningful but somewhat interpretable), supports supervised / semi-supervised variants, can transform new points (t-SNE cannot). Both are for visualization primarily.

UMAP key hyperparameters.

(1) $n_{\mathrm{neighbors}}$: local vs global (5-15 local, 50-100 global; default 15). (2) $\mathrm{min_{dist}}$: minimum spacing in embedding — small (0.1) preserves clusters, large (0.5) spreads points.

Isomap — what does it do?

Non-linear DR that preserves geodesic (manifold) distances. Steps: (1) k-NN graph, (2) all-pairs shortest paths (Dijkstra / Floyd-Warshall) → geodesic distance matrix, (3) MDS on geodesic distances → low-dim embedding.

Locally Linear Embedding (LLE) — how does it work?

(1) Find k nearest neighbors for each point. (2) Reconstruct each point as linear combination of neighbors: minimize $\|x_i\ -\ \Sigma\ w_{\mathrm{ij}}\ x_j\|^2$ with Σ $w_{\mathrm{ij}} = 1$.

Multidimensional Scaling (MDS) — variants.

Given pairwise distance matrix D, find low-dim embedding preserving distances. Classical MDS: closed form via eigendecomposition of doubly-centered D^2 — equivalent to PCA on distance data.

Autoencoder for dimensionality reduction — pros and cons.

Pros: non-linear, scalable (SGD), transformable (new data → embedding), can be regularized (denoising, sparse, contractive). Cons: no orthogonality (hard to interpret), no explained-variance ratios, sensitive to hyperparameters, harder than PCA.

What is the manifold hypothesis?

Real high-dimensional data (images, audio, text) lies on a low-dimensional manifold embedded in the high-dim space — e.g. natural images inhabit ~O(hundreds)-dim manifold within millions of pixels. Motivates non-linear DR (Isomap, UMAP, autoencoders): find that low-dim coordinate system.

How do you estimate the intrinsic dimension of a dataset?

(1) Correlation dimension (Grassberger-Procaccia): slope of $\log(\mathrm{number\ of\ pairs\ within}\ \varepsilon)$ vs log ε. (2) MLE-based (Levina-Bickel): from distances to k nearest neighbors.

Truncated SVD vs PCA on sparse data.

PCA subtracts the mean → densifies sparse matrices (bad for TF-IDF, one-hot). Truncated SVD (scikit-learn's TruncatedSVD, also known as LSA in text) does SVD without centering → preserves sparsity, scales to millions of features.

Incremental PCA — when do you need it?

Dataset doesn't fit in RAM. Process mini-batches, update running estimate of principal components (block Lanczos or Ross et al.'s method). scikit-learn: IncrementalPCA.

Robust PCA — what problem does it solve?

Standard PCA is sensitive to outliers (single bad row can rotate components). Robust PCA (Candès et al.): decompose X = L + S where L is low-rank (clean data) and S is sparse (outliers).

Why does truncated SVD denoise?

Signal typically lies in low-rank subspace; noise spreads across all singular directions. Discarding small singular values throws away noise-dominant components while keeping signal.

Word embeddings as unsupervised DR of text.

Word2Vec (skip-gram, CBOW), GloVe: predict context words → learn dense word vectors capturing co-occurrence structure. Word2Vec is implicit PMI matrix factorization (Levy & Goldberg 2014).

How do modern sentence / doc embeddings work?

Sentence-BERT (Reimers-Gurevych): fine-tune BERT with siamese contrastive loss on paraphrase / NLI pairs → semantically-meaningful vectors, cosine similarity ≈ semantic similarity. Modern: E5, BGE, GTE, OpenAI text-embedding-3, Cohere embed-v3 — trained on contrastive tasks over huge multilingual corpora.

Matrix completion — how does it relate to unsupervised learning?

Recover missing entries in a matrix by assuming low-rank structure. Solve min $\text{rank}(M)$ s.t. observed entries match — NP-hard, relaxed to min ||M||_* (nuclear norm).

Principal Components Regression (PCR) — what does it do?

Regress on PCA components instead of raw features: (1) PCA → keep top k components, (2) OLS on those. Reduces multicollinearity + acts as regularization.

Partial Least Squares (PLS) — how is it different from PCR?

PCR: PCA components chosen by max variance in X (Y-ignorant). PLS: components maximize covariance with Y (supervised DR).

LSA (Latent Semantic Analysis) — how does it relate to modern retrieval?

Truncated SVD on TF-IDF document-term matrix. Rows → topic vectors for docs, columns for terms.

Matrix factorization for recsys — objective.

Model rating $r_{ui} \approx p_u' * q_i$ where $p_u \in R^k$ is user vector, q_i is item vector. Minimize $\Sigma\ (r_{ui} - p_u q_i)^2 + \lambda\ (||\ p_u\ ||^2 + ||\ q_i\ ||^2)$ over observed entries.

Interview: 'you have 500 features, most correlated — how do you preprocess?'

(1) Correlation heatmap → drop obviously redundant. (2) Domain-driven grouping (aggregate related features first).

Interview: 'when should you use PCA vs autoencoder for dim reduction?'

PCA: (1) linear structure, (2) < 200 features, (3) want interpretability (loadings), (4) fast + closed form, (5) baseline. Autoencoder: (1) non-linear structure (images, sequences), (2) very high dim (thousands+), (3) enough data to train, (4) willing to accept less interpretability, (5) transformable to new points, (6) regularizable (denoising, sparsity).

How many principal components do you keep, and what does 95% variance actually guarantee?

Common choices are a cumulative variance threshold, the knee in the scree plot, or the number that maximizes downstream performance, and the last is the only one tied to your goal. What a 95% variance threshold guarantees is only that reconstruction error in the least-squares sense is small; it says nothing about whether the discarded 5% contained the signal you care about.

What conclusions can you not draw from a t-SNE plot?

Cluster sizes are meaningless, because t-SNE expands sparse regions and compresses dense ones to fit everything into two dimensions, so a visually large blob is not a numerous group. Distances between clusters are also unreliable: the method preserves local neighbourhoods and deliberately sacrifices global geometry, so two well-separated blobs may be closer in the original space than they look.

Anomaly & outlier detection

What are the main approaches to anomaly detection?

(1) Statistical: fit a distribution and flag low-density points (z-score, Gaussian mixtures). (2) Distance/density-based: LOF, DBSCAN.

Isolation Forest — how does it detect anomalies?

Build many random trees splitting on random features and thresholds. Anomalies are 'isolated' with fewer splits than inliers (shorter path length).

One-class SVM — mechanism and pitfalls.

Learns a decision boundary enclosing 'normal' data by maximizing margin from origin in RKHS (with RBF kernel typically). Parameter ν upper-bounds fraction of outliers.

Autoencoder for anomaly detection — how?

Train AE on 'normal' data — it learns to reconstruct normal patterns well. At inference, compute reconstruction error ||x - x̂||²; high error = anomaly.

VAE-based anomaly detection — advantages.

VAE learns probabilistic latent space with p(x). Anomaly score: (1) reconstruction error, (2) negative ELBO, (3) KL between $q(z \mid x)$ and prior.

Deep SVDD — the core idea.

Deep Support Vector Data Description (Ruff et al. 2018): train a neural encoder φ_θ to map all normal training data into a small hypersphere in feature space (center c, minimum radius). Anomaly score: distance || φ_θ(x) - c||.

Semi-supervised vs unsupervised anomaly detection.

Unsupervised: no labels; assumes normals dominate. Semi-supervised: only normal data labeled — train on clean normals, detect deviations (one-class SVM, deep SVDD, AE).

Positive-Unlabeled (PU) learning — when useful?

You have some confirmed positives (frauds, anomalies) and a large unlabeled pool that mixes positives + negatives. Standard classification is biased.

Anomaly detection in time series — what changes?

Temporal context matters. Options: (1) statistical: rolling mean/std + z-score, ARIMA residuals, STL decomposition + IQR on residuals.

How is drift detection an unsupervised problem?

Compare current feature distributions to training / baseline distributions without labels. Methods: (1) KS test per feature (univariate).

Maximum Mean Discrepancy (MMD) — what is it?

Distance between two distributions in RKHS: MMD(P, Q) = ||$\mu_P - \mu_Q$||_H where μ_P is the mean embedding. Estimated from samples using kernel k (Gaussian typical): $\mathrm{MMD}^2 = E[k(x, x)] + E[k(y, y)] - 2$ E[k(x, y)].

Wasserstein distance — intuition.

'Earth mover's distance': minimum cost of transporting mass from P to Q where cost = distance × mass. $W_1(P, Q) = \inf\_\gamma$ E_γ[|| $x - y$ ||] over couplings γ.

Covariate drift vs concept drift vs label drift — the differences.

Covariate drift: P(X) changes, $P(Y \mid X)$ same — model still valid, just used on different inputs (retrain if severe). Concept drift: $P(Y \mid X)$ changes — model relationship broken, must retrain.

Out-of-distribution (OOD) detection — approaches.

Softmax confidence is unreliable (over-confident on OOD). Better: (1) MSP with temperature scaling, (2) energy score (LeCun et al.), (3) ODIN (temperature + input perturbation), (4) Mahalanobis distance in feature space, (5) deep generative likelihood (with Nalisnick's caveat), (6) contrastive OOD detectors.

Conformal prediction for anomaly detection.

Given calibration set of normal points, use non-conformity score to produce p-values with guaranteed (1-α) coverage on inliers under exchangeability. p-value < α → outlier at level α. Distribution-free, model-agnostic → wraps any anomaly detector for calibrated alarms.

Scan statistics — when do you use them?

Detect unusual clusters in space-time (disease outbreaks, crime hotspots, network intrusion). Slide a window over spatial / temporal regions, compare observed vs expected counts (usually Poisson).

CUSUM change-point detection — how does it work?

Cumulative sum of deviations from a reference mean: $S_t = \max(0, S_{t-1} + (x_t - \mu) - k)$. Alarm when $S_t > \mathrm{threshold}$ h.

How do you handle 99.9% normal / 0.1% anomaly training data?

(1) Semi-supervised: train only on normal (assume clean). (2) Isolation Forest / LOF: designed for exactly this ratio.

How do you explain why a point is anomalous?

(1) Per-feature contribution: which feature's z-score / reconstruction error / SHAP value drove the anomaly. (2) Nearest-normals: 'this looks unusual because normally X, Y differ by ...'.

Interview: 'design an unsupervised fraud detection system'.

(1) Feature engineering: transaction amount, velocity, device / IP fingerprints, merchant risk, time-of-day, historical account patterns. (2) Stack detectors: (a) Isolation Forest on tabular features, (b) autoencoder on account behavior sequences, (c) graph-based (transaction network) for money-laundering rings.

How would you detect anomalies in multi-modal data (image + tabular)?

(1) Fuse: (a) tabular through MLP encoder, (b) image through CNN encoder → concat embeddings → joint autoencoder or joint density model. (2) Score per modality then combine (max or weighted sum).

How would you detect fraud rings (colluding accounts)?

(1) Build interaction graph: accounts as nodes, shared devices/IPs/transactions as edges. (2) Community detection (Louvain, Leiden, HDBSCAN on node embeddings).

Interview: production fraud detector — daily volume 10M, current FN too high.

(1) Investigate: what patterns are FNs? Feature gaps?

Interview: model trained in region A must now serve region B — approach?

(1) Diagnose: KS + PSI feature-by-feature; MMD/classifier-drift for multivariate. (2) If severe drift, retrain on B data (labeled ideally).

Interview: 'design anomaly detection for a factory sensor with 200 signals.'

(1) EDA: distribution + autocorrelation per signal; note skew / seasonality. (2) Feature engineering: rolling stats + FFT / wavelet features + inter-sensor correlations.

Isolation forest or autoencoder for anomaly detection?

Isolation forest for tabular data, as a first attempt in nearly every case: it is fast, needs almost no tuning, handles mixed scales tolerably, and its notion of an anomaly as a point that is easy to isolate is easy to explain to a stakeholder. Autoencoders earn their complexity when anomalies are defined by structure a tree cannot see, such as images, spectrograms, or long sequences where the relationship between dimensions is the signal.

Autoencoders & VAEs

What is self-supervised learning and why does it matter?

Self-supervised learning creates labels from the data itself via pretext tasks (masked language modeling, next-token prediction, contrastive views of images) — no manual annotation. It matters because it lets us pretrain huge models on unlabeled data and then transfer to downstream tasks with little labeled data.

How does contrastive learning work?

Create two augmented views of the same example (positive pair) and treat other examples as negatives. Train an encoder so positives are close in embedding space and negatives are far, using a loss like InfoNCE.

What is an autoencoder and what are the common variants?

An autoencoder learns to compress input x through a bottleneck and reconstruct it. Variants: denoising AE (train to reconstruct clean from noisy input), sparse AE (bottleneck via sparsity penalty), variational AE (probabilistic latent, generative), masked AE (mask patches, reconstruct — used in vision pretraining).

Autoencoder variants — quick tour.

(1) Undercomplete AE: bottleneck < input dim → compression. (2) Denoising AE: reconstruct clean from noised input → robustness / representation.

Derive the VAE ELBO.

log p(x) = log ∫ $p(x \mid z)$ p(z) dz ≥ $E_{q(z|x)}[\log\ p(x \mid z)] - \mathrm{KL}(q(z \mid x)\ ||\ p(z))$ (Jensen). ELBO decomposes into reconstruction (log-likelihood of x given z) minus KL to prior.

VAE vs plain autoencoder — the key advantages.

(1) Probabilistic latent → generative (sample z ~ p(z), decode → new x). (2) Regularized latent space by KL to prior → smooth interpolation, meaningful arithmetic.

β-VAE — what does the β hyperparameter do?

Modifies ELBO to $E_q[\log\ p(x \mid z)]\ -\ \beta\ \cdot\ \mathrm{KL}(q\ ||\ p)$. β > 1 pushes KL harder → more independent latent factors → disentangled representations (each latent dim captures one factor: pose, color, size). Cost: worse reconstruction.

VQ-VAE — the core idea.

Discrete latent codes: encoder output quantized to nearest code in a learned codebook (like k-means). Straight-through gradient estimator for backprop.

SimCLR — recipe.

Contrastive framework for visual SSL: (1) two random augmentations of each image → positive pair, (2) encoder + projection head, (3) NT-Xent loss (InfoNCE variant): pull positives together, push all other batch items apart. Needs LARGE batch size (~4096) for enough negatives.

MoCo (Momentum Contrast) — how does it enable smaller batches?

Maintain a queue of negatives from previous batches instead of relying on current batch only. Momentum encoder: EMA of the main encoder → consistent keys in queue as encoder updates slowly.

BYOL — how does it avoid the need for negatives?

Two networks: online (learnable) and target (EMA of online). Feed different augmentations to each; online must predict target's representation via a predictor head.

SimSiam — what makes it minimal?

Removes EMA target: both branches share weights. Uses only stop-gradient on one branch and an asymmetric predictor head.

DINO — self-distillation with no labels.

Student network learns to match teacher (EMA of student) on different augmentations of same image. Uses ViT backbone → produces amazing self-attention maps (unsupervised object segmentation emerges!).

MAE (Masked Autoencoder) — He et al. 2022.

Mask 75% of image patches, encode only visible patches, decode all patches (mask tokens). Loss: MSE on masked pixel patches.

CLIP — how does contrastive image-text training work?

Train two encoders (image ViT + text Transformer) so that matching (image, caption) pairs are close in shared embedding space, mismatched pairs far. InfoNCE loss over batch.

JEPA (I-JEPA, V-JEPA) — LeCun's alternative to generative SSL.

Predict abstract features (not pixels) of masked regions in embedding space. Sidesteps pixel-level detail (irrelevant + wasteful) and models context predictively.

InfoNCE loss — formula and intuition.

L = -log $[\exp(s(x,\ x^{+})/\tau)\ /\ \Sigma_j\ \exp(s(x,\ x_j)/\tau)]$ — softmax over one positive and many negatives, temperature τ. Cross-entropy that treats classification 'which of N is the positive?'.

Why are augmentations so important in contrastive SSL?

Augmentations define the invariances the encoder learns — 'same-object under transformation' becomes the training signal. Strong compositions (crop + color jitter + Gaussian blur) work best (Chen et al. 2020).

Word2Vec skip-gram — objective and training.

For each target word, predict surrounding context words (window size ~5). Softmax over full vocabulary too expensive → negative sampling: for each positive (target, context) pair, sample k random 'negative' words and train binary logistic.

GloVe vs Word2Vec.

GloVe (Pennington 2014): explicit weighted matrix factorization of log co-occurrence counts. Objective: $\log(X_{\mathrm{ij}}) \approx w_i{}'\ w_j\ +\ b_i\ +\ b_j$.

Word embedding analogies — why do they work?

king - man + woman ≈ queen. Emerges because embeddings capture distributional differences.

Sentence-BERT — how does it improve on BERT for retrieval?

BERT's [CLS] token is not a great semantic sentence vector out of the box. SBERT fine-tunes BERT with a siamese architecture on NLI + STS: same encoder on two sentences, minimize distance for paraphrases, maximize for contradictions.

Node2Vec — how does it learn graph node embeddings?

Random walks starting from each node → treat walks as 'sentences', apply Word2Vec skip-gram → embed nodes. Biased walks (p, q) control breadth (BFS-like) vs depth (DFS-like): controls whether embeddings capture structural equivalence or community proximity.

Modern graph representation learning — GNNs.

Graph Neural Networks aggregate neighbor features iteratively: $h_v^{l+1}\ =\ \mathrm{UPDATE}(h_v^l,\ \mathrm{AGG}(h_u^l\ \ u\ \in\ N(v)))$. Variants: GraphSAGE (sampled aggregation), GAT (attention weights), GCN (spectral), MPNN (message passing).

Self-supervised graph learning — approaches.

(1) Node-level: contrastive on augmented views (drop edges/nodes/features), GraphCL. (2) Predict masked node attributes (MaskGAE).

Self-supervised audio — wav2vec 2 and HuBERT.

Wav2Vec 2 (Facebook 2020): quantize audio features → mask spans → predict quantized targets via contrastive loss. HuBERT: similar but uses clustered representations as pseudo-labels (BERT-style masked prediction).

How are multimodal embeddings unified across text / image / audio?

Approaches: (1) CLIP-style paired training (image ↔ text). (2) ImageBind (Meta 2023): 6 modalities aligned through image as the 'bridge' — no need for all-pair data.

Scaling laws for self-supervised pretraining.

Downstream performance improves as a power law in (data, params, compute). MAE / CLIP / DINOv2 / vision transformers all show emergent capabilities at scale (rare classes, zero-shot).

Mode / representation collapse in SSL — what and why?

Encoder outputs the same (or trivially degenerate) representation for all inputs → useless embeddings. Root cause: shortcut solutions minimizing loss without capturing content.

VICReg / Barlow Twins — non-contrastive SSL via covariance regularization.

Barlow Twins: cross-correlation of two augmented view embeddings should be identity (diagonal = 1, off-diag = 0 → invariance + de-correlation). VICReg extends with three terms: invariance (MSE between views), variance (each dim has SD > threshold), covariance (off-diag = 0).

Fine-tuning vs linear probe vs prompt-tuning — when do you use each?

(1) Linear probe: small labeled data + strong SSL encoder → freeze encoder, add linear head. Fast, tiny compute.

What is a 'foundation model' in the unsupervised sense?

Large model pretrained on massive unlabeled data via SSL → generalizes to many downstream tasks with little or no fine-tuning. Examples: GPT / Llama (text), CLIP / DINOv2 (vision), SAM (segmentation), Whisper (speech), ImageBind (multimodal).

Interview: 'you have unlabeled images — which SSL method should you use?'

Ask: (1) Compute budget? MAE is fastest to train.

Interview: 'when should you use a VAE vs GAN vs diffusion for generation?'

(1) VAE: fast, latent space useful for interpolation + representation, but blurry samples. Best when interpretable latent matters (drug design, molecule generation).

Topic modeling & text

What is topic modeling and when do you use LDA?

Topic modeling discovers latent themes in a document corpus. LDA (Latent Dirichlet Allocation) models each document as a mixture of topics and each topic as a distribution over words.

TF-IDF weighting — formula and rationale.

TF: term frequency in document (raw, log, or sublinear). IDF: $\log(N \ / \ \mathrm{df}_t)$ — downweights common words appearing in many documents.

BM25 — why is it still competitive with modern retrievers?

Okapi BM25 = probabilistic TF-IDF with length normalization + tunable k_1 (TF saturation) + b (length norm). Very fast (inverted index), well-understood, competitive on exact-match / keyword-heavy queries where dense embeddings miss.

Vector search / approximate nearest neighbors — main algorithms.

(1) HNSW (Hierarchical Navigable Small World): graph-based, dominant in industry (Qdrant, Weaviate, pgvector). (2) IVF-PQ (inverted file + product quantization): FAISS default for very large indexes.

How is RAG retrieval an unsupervised problem?

RAG (retrieval-augmented generation) retrieves passages from a corpus based on unsupervised embeddings (no labels of 'relevant' pairs). Uses: dense retrieval (sentence embeddings + cosine + ANN), often hybrid with BM25.

PMI and PPMI — what they measure.

PMI(x, y) = log[P(x, y) / (P(x) P(y))]. Positive → co-occur more than random.

How do you cluster / retrieve code snippets?

(1) Code-specific embeddings: CodeBERT, StarCoder, GTE-code, OpenAI text-embedding-3 handle code decently. (2) Combine with AST features / API signatures.

Interview: users complain search returns irrelevant results — how do you fix?

(1) Analyze query logs: are they specific / broad / navigational / typos? (2) Baseline: BM25 lexical.

Interview: choose a sentence embedding model for a startup RAG.

Constraints: quality vs cost vs latency vs vector dimension. (1) OpenAI text-embedding-3-small: cheap ($0.02/1M), 1536 dim, hosted.

Interview: 'you have 1M support tickets — how do you categorize them?'

(1) Deduplicate near-identical tickets (LSH). (2) Sentence-transformer embed (bge-small or multilingual if needed).

Recommender & association

Collaborative filtering — how does it use unsupervised methods?

Learn user + item embeddings from interaction matrix (implicit or explicit ratings). No content features required — 'unsupervised' in the sense that only interactions are used.

Implicit feedback vs explicit ratings in recsys.

Explicit: users give a star rating. Rare in practice.

How do you handle the cold-start problem?

New user or item with no interactions. Solutions: (1) Content-based features (user profile, item text/image → embed) → hybrid recsys.

Two-tower recsys — architecture and use.

User tower: NN encoding user features / history → user embedding. Item tower: NN encoding item features → item embedding.

Association rule mining — Apriori & FP-Growth.

Find frequent itemsets in transactions (e.g. 'diapers → beer'). Apriori: level-wise search, prune non-frequent supersets — slow, many DB scans.

How does image similarity search work in production?

(1) Extract embeddings: CLIP / DINOv2 / SigLIP → 512-1024 dim vectors. (2) L2-normalize + index in FAISS / Qdrant / pgvector (HNSW / IVF-PQ).

Market basket analysis — modern approach.

Classical: association rules (Apriori/FP-Growth). Modern: (1) product embeddings via word2vec-on-baskets ('prod2vec'), (2) session-based recsys (GRU4Rec, SASRec) on sequential purchases, (3) graph embeddings on co-purchase network, (4) contextual bandits for personalization.

Unsupervised image segmentation — approaches.

(1) Classical: k-means or mean-shift on pixel color + position (SLIC superpixels). (2) Spectral clustering on affinity graph.

Zero-shot image classification via CLIP — mechanism.

(1) Encode candidate class names as text ('a photo of a dog', 'a photo of a cat') → text embeddings. (2) Encode query image → image embedding.

How does unsupervised learning help with noisy labels?

(1) Cluster the data unsupervisedly; check consistency of labels within clusters — outliers likely mislabeled. (2) Confidence learning (Northcutt et al., cleanlab): use self-consistency + probabilistic label pruning.

Interview: design a recommender for a new streaming service.

(1) Cold-start heavy: no history. Start with content-based recsys (title/description embeddings, genre, actors → cosine similarity to what user selects).

Interview: your model's accuracy dropped 15% overnight — how do you diagnose?

(1) Rule out incidents first: pipeline break, feature-source outage, label logging bug. (2) Check input distributions: KS + PSI per feature vs baseline.

Interview: you must ship an embedding service serving 100M vectors, 10ms p95 latency.

(1) Choose embedding model by quality + latency (E5-small / bge-small / OpenAI text-embedding-3-small). (2) Batch-embed the corpus offline.

Interview: how do you monitor drift in an embedding-based retrieval system?

(1) Query distribution drift: KS/PSI on query embedding norms, cluster distribution of queries over time. (2) Corpus drift: same on doc embeddings.

Interview: 'when should you use graph-based methods over tabular?'

(1) Data has natural relational structure (users↔items, molecules, transactions, social nets). (2) Prediction depends on neighborhood (link prediction, fraud rings, drug interactions).

Interview: 'you have to embed 100M documents monthly — cost strategy?'

(1) Only re-embed changed / new documents (change-data-capture). (2) Choose model by cost/quality tradeoff: OpenAI-3-small ($0.02/1M tokens) vs OSS bge-small hosted on your infra (initial GPU spend, then free).

Interview: 'summarize when unsupervised learning wins in production.'

Unsupervised wins when: (1) labels are expensive / impossible (fraud rings, anomalies, cold-start recsys). (2) Data has natural structure worth exposing (customer segments, log templates, image duplicates).

How do you make recommendations for a brand-new user?

Fall back through a ladder of decreasing personalization. With no history, serve popularity, ideally popularity within whatever context you do know, such as country, device, or referral source, which is far better than global top items.