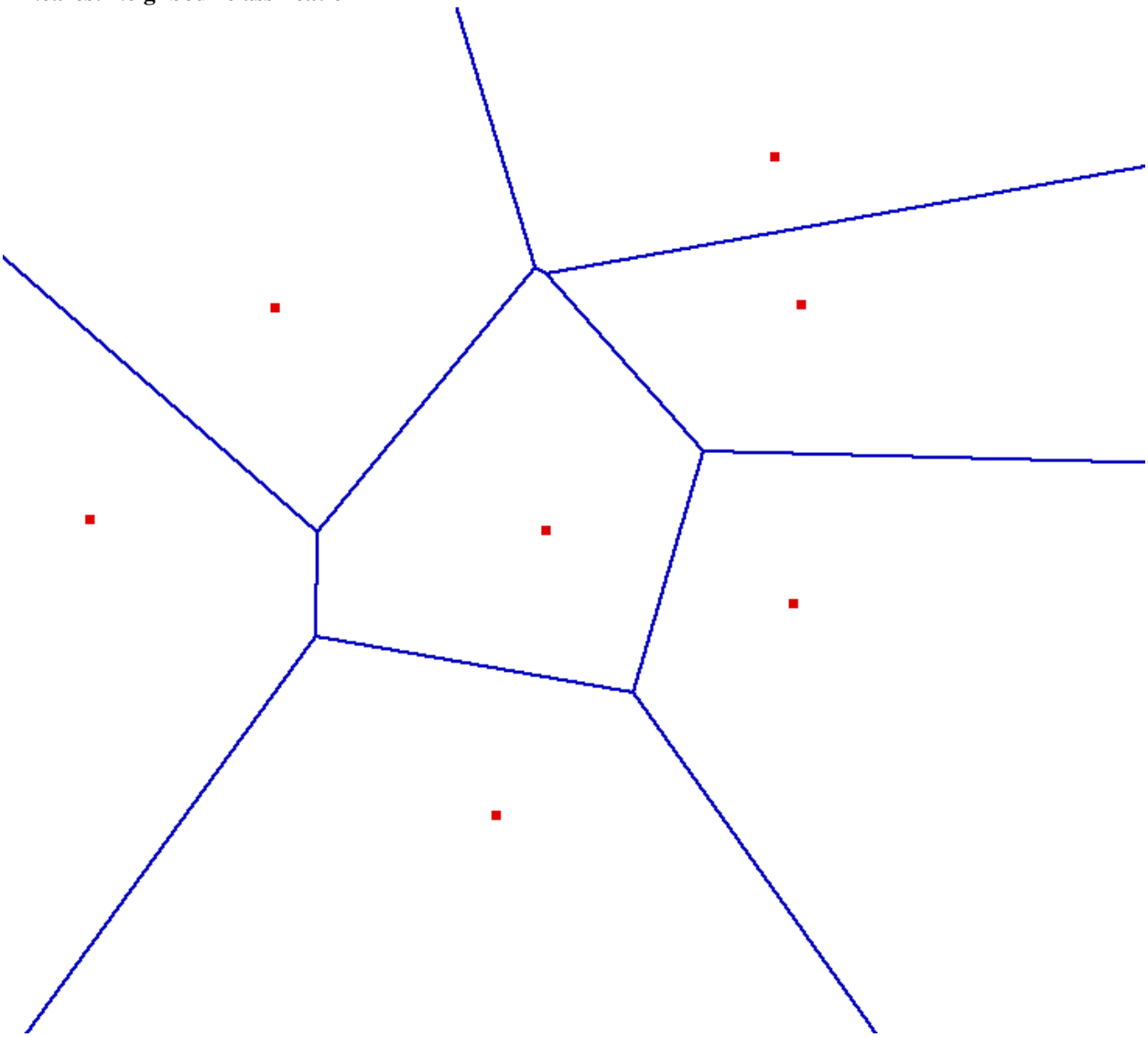
1-Nearest Neighbour classification



Prediction rule

Assign a query to the class of its closest training example:

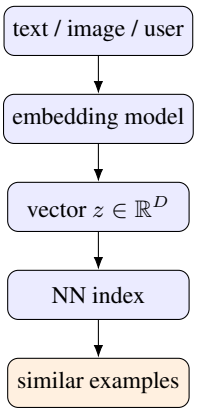
$$\hat{y}(x) = y_{i^*}, \quad i^* = \arg\min_i d(x, x_i).$$

- No explicit parametric model is fitted.
- The decision boundary is induced by the stored examples.
- For Euclidean distance, 1-NN regions form a Voronoi tessellation.

Why study nearest neighbours today?

- Simple non-parametric baseline

- Naturally nonlinear and multiclass
- Supports classification, regression, retrieval, and anomaly detection
- Predictions can be explained through retrieved examples



Modern perspective
Many contemporary systems first map objects to vectors and then perform nearest-neighbour search in the learned embedding space.

Measuring the distance between instances

Metric
A function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$ is a metric if, for all $x, y, z \in \mathcal{X}$,

$$d(x, y) = 0 \Leftrightarrow x = y, \qquad d(x, y) = d(y, x), \qquad d(x, z) \leq d(x, y) + d(y, z).$$

Minkowski distances

$$d_p(x, y) = \left(\sum_{j=1}^{D} |x_j - y_j|^p \right)^{1/p}.$$

$p = 1$: Manhattan; $p = 2$: Euclidean.

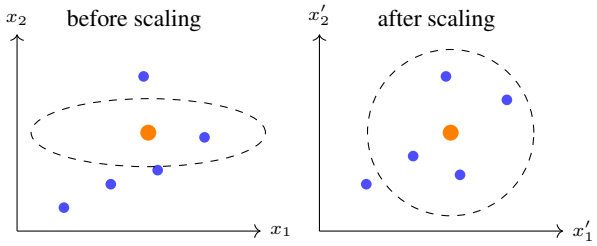
Other useful choices

- cosine distance for directions/embeddings;
- Hamming distance for binary attributes;
- Mahalanobis distance for correlated features.

Feature scaling changes neighbourhoods
With Euclidean distance, a feature with a large numerical range can dominate all others.

- Standardization: $x_j \leftarrow (x_j - \mu_j)/\sigma_j$
- Robust scaling for heavy-tailed features
- Normalize vectors before cosine similarity

- Estimate preprocessing on training data only



Practical lesson
The metric and the representation are part of the model, not neutral preprocessing choices.

k-Nearest Neighbour classification

Algorithm
For each query x:

1. compute $d(x, x_i)$ for all training examples;
2. select the indices $N_k(x)$ of the k smallest distances;
3. return the majority class:

$$\hat{y}(x) = \arg\max_y \sum_{i \in N_k(x)} \mathbb{I}[y_i = y].$$

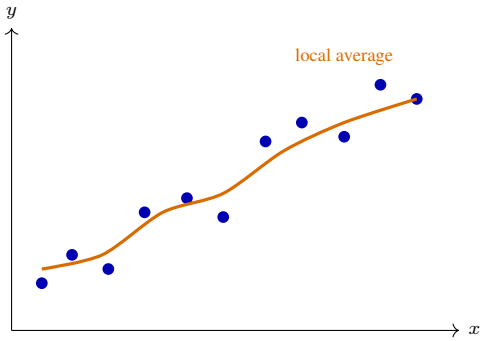
- Odd k reduces ties in binary classification.
- Class probabilities can be estimated from neighbour frequencies.

k-Nearest Neighbour regression

Local averaging

$$\hat{y}(x) = \frac{1}{k} \sum_{i \in N_k(x)} y_i.$$

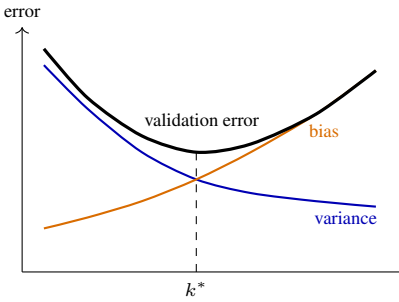
- Small k: flexible but noisy estimate
- Large k: smoother but potentially biased
- Predictions remain within the range of nearby observed targets
- Poor extrapolation outside the training support



Choosing k: a bias–variance trade-off

Small k Low bias, high variance; sensitive to noise.

Large k Higher bias, lower variance; smoother decision boundary.



Select k using validation or cross-validation, together with the distance metric and weighting rule.

Distance-weighted and radius-based neighbours

Distance weighting

$$\hat{y}(x) = \arg\max_y \sum_{i \in N_k(x)} w_i \mathbb{I}[y_i = y],$$

with

$$w_i = \frac{1}{(d(x, x_i)}$$

For regression:

$$\hat{y}(x) = \frac{\sum_i w_i y_i}{\sum_i w_i}.$$

Radius neighbours
Use every training example satisfying

$$d(x, x_i) \leq r.$$

- Adapts the number of neighbours to local density.
- Requires a policy when no point lies within radius r.

Characteristics of nearest-neighbour learning

Instance-based Predictions are based directly on stored examples.

Lazy Model fitting is cheap; most computation occurs at query time.

Local The prediction assumes nearby examples have similar outputs.

Non-parametric Complexity can increase with the amount of data.

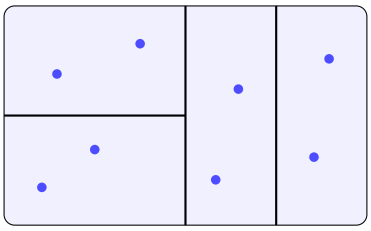
Representation-sensitive Irrelevant or badly scaled features distort neighbourhoods.

Computational baseline
With n examples and D features, brute-force prediction requires $O(nD)$ work per query and $O(nD)$ storage.

Exact nearest-neighbour search

Brute force
Compute every distance. Simple, exact, parallelizable, and often competitive for small datasets or dense vector batches.

KD tree
Recursively partitions space with axis-aligned cuts. Effective mainly in low dimension.



spatial partition and pruning

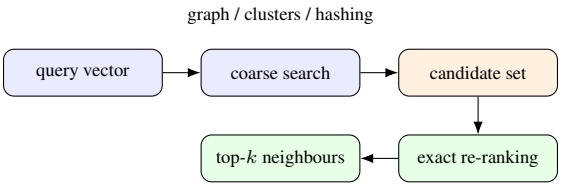
Approximate nearest-neighbour search
For large, high-dimensional datasets, exact search may be too slow or memory intensive.

Graph indexes HNSW navigates a hierarchy of proximity graphs.

Inverted indexes IVF searches only selected coarse clusters.

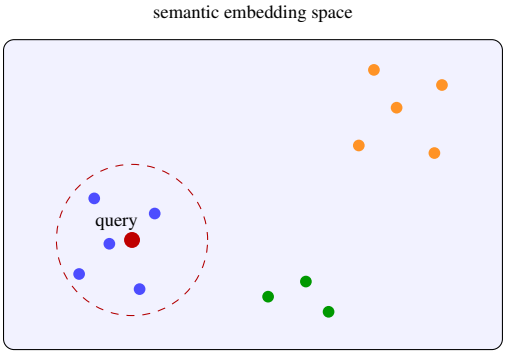
Quantization Product quantization compresses vectors and approximates distances.

Hashing LSH maps likely neighbours to shared buckets.



Nearest neighbours in embedding spaces
Images, text, users, and products can be represented by learned vectors $z = f_\theta(x)$.

- semantic retrieval;
- recommendation;
- duplicate detection;
- retrieval-augmented generation;
- few-shot classification by prototypes or exemplars.



Important
Efficient search cannot repair a poor representation: useful neighbours require an embedding and similarity measure aligned with the task.

Learning the distance

Uniform feature weighting is often unrealistic. A parameterized distance can be learned from labelled data:

$$d_M(x, x') = \sqrt{(x - x')^\top M (x - x')}, \qquad M \succeq 0.$$

- diagonal M: feature weighting;
- full M: rotations and correlated directions;
- neural embedding f_θ: nonlinear metric learning.

Training signal

Pull similar examples together and push dissimilar examples apart, using pairs, triplets, or neighbourhood-based objectives.

Practical workflow

1. Define the task-appropriate representation and distance.
2. Fit scaling, imputation, and embedding transformations on training data only.
3. Tune k, weighting, and distance by validation.
4. Start with vectorized brute force as a correctness baseline.
5. For low-dimensional exact search, benchmark KD trees.
6. At large scale, select an appropriate approximate nearest-neighbour search index.

Take-home messages

1. k-NN is a local, instance-based, non-parametric learning method.
2. Its behaviour is controlled by the representation, metric, k, and weighting rule.
3. Scaling and irrelevant dimensions can radically change neighbourhoods.
4. Exact search ranges from brute force to metric and spatial trees.
5. Approximate indexes trade a small amount of recall for major gains in latency and memory.
6. Modern retrieval systems frequently combine learned embeddings with nearest-neighbour search.