Instance-Based vs Model-Based Learning in Machine Learning

Overview — Core Distinction

- **Instance-based learning (IBL)**: the learner *stores training instances* and defers generalization until query time. It is "lazy": little or no work at training, heavy work at prediction. Examples: k-nearest neighbors (k-NN), kernel regression, locally weighted regression, case-based reasoning.
- **Model-based learning (MBL)**: the learner *builds an explicit model* (a function or parameters) from the training data, then uses that compact model to answer queries. It is "eager": heavy training, fast inference. Examples: linear/logistic regression, SVM, decision trees, neural networks, Gaussian processes.

There exists a spectrum rather than a strict binary. For example, SVMs in dual form depend on support vectors (a subset of instances) — they are hybrids. Gaussian processes also store covariance across points. Always think in terms of:

(a) where/when the work happens (train vs predict),

(b) what is stored (raw examples vs parameters).

Key Axes of Comparison

- Lazy vs Eager generalization
- Memory usage
- Training vs Prediction cost
- Parametric vs Nonparametric
- Local vs Global generalization
- Sensitivity: noise, feature scaling, dimensionality
- Update pattern

Instance-Based Learning (IBL) — Deep Dive

Intuition

Prediction for a query x is made by looking at stored examples $\{(x_i, y_i)\}_{i=1}^n$ and combining the labels/values of nearby examples in feature space. "Nearby" is defined by a distance or similarity metric.

Core Algorithms and Formulas

k-Nearest Neighbors (Classification).

$$\hat{y}(x) = \text{mode}\{y_i : i \in N_k(x)\}.$$

Weighted voting:

$$\hat{y}(x) = \arg\max_c \sum_{i \in N_k(x)} w(d(x, x_i))\, \mathbf{1}\{y_i = c\},$$

where $w(r) = \exp\left(-\frac{r^2}{2h^2}\right)$ or $w(r) = 1/r$.

k-Nearest Neighbors (Regression).

$$\hat{y}(x) = \frac{1}{\sum_{i \in N_k(x)} w_i} \sum_{i \in N_k(x)} w_i y_i, \quad w_i = K\Big(\frac{d(x, x_i)}{h}\Big).$$

Kernel / Nadaraya–Watson Estimator.

$$\hat{y}(x) = \frac{\sum_{i=1}^n K\left(\frac{d(x,x_i)}{h}\right) y_i}{\sum_{i=1}^n K\left(\frac{d(x,x_i)}{h}\right)}.$$

Locally Weighted Linear Regression (LWR).

$$\hat{\beta}(x) = (X^T W X)^{-1} X^T W y, \quad \hat{y}(x) = x^T \hat{\beta}(x).$$

Distance and Similarity

- Continuous numeric: Euclidean, Manhattan, Mahalanobis, cosine.
- Categorical: Hamming, overlap, or learned embeddings.
- Feature scaling is **mandatory**.

Computational Complexity

- Training: $O(1)$ (store dataset).
- Prediction: Naïve $O(nd)$, with KD-tree $O(\log n)$ (low-d).

Strengths

- Simple, interpretable locally.
- No training step.
- Handles complex decision boundaries.
- Theoretically consistent (approaches Bayes error).

Weaknesses

- High prediction cost.
- Curse of dimensionality.
- Sensitive to irrelevant features.
- Memory heavy.

Improvements

Prototype selection, metric learning, approximate nearest neighbor search (ANN), dimensionality reduction, hybrid embeddings + kNN.

Applications

Recommender systems, anomaly detection, case-based reasoning, few-shot learning.

Model-Based Learning (MBL) — Deep Dive

Intuition

Learn a function f_θ with parameters θ from data. Training is computation-heavy; inference is fast.

Examples

- Parametric: linear regression, logistic regression, neural networks.
- Nonparametric: decision trees, Gaussian processes.
- Generative vs Discriminative models.

Mathematical Formulations

Linear Regression (Ridge).

$$\hat{\beta} = (X^T X + \lambda I)^{-1} X^T y.$$

Logistic Regression.

$$\min_w \; -\sum_i \Big[y_i \log \sigma(w^T x_i) + (1 - y_i) \log(1 - \sigma(w^T x_i)) \Big] + \lambda \|w\|^2.$$

Neural Networks. Loss optimization via gradient descent.

Gaussian Processes. Require inversion of K_{nn}, cost $O(n^3)$ training.

Strengths

- Fast inference, compact storage.
- Learns complex feature representations.
- Can incorporate priors.
- Probabilistic outputs.

Weaknesses

- High training cost.
- Need for large datasets.
- Risk of model misspecification.
- Interpretability varies.

Practical Tips

- Start with simple models.
- Regularize properly.
- Tune hyperparameters with cross-validation.
- Use incremental algorithms for streaming data.

Hybrids and Connections

- Kernel methods / SVMs depend on support vectors.
- RBF networks combine prototypes with parametric learning.
- Gaussian processes are model-based but nonparametric.
- Embedding + kNN (e.g., Siamese nets).

Bias–Variance and Smoothing View

- Small k in kNN $\Rightarrow$ low bias, high variance.
- Large $k \Rightarrow$ high bias, low variance.
- In parametric models, increasing complexity reduces bias but increases variance.

Checklist: How to Choose

1. Dataset size.
2. Dimensionality.
3. Latency requirements.
4. Update frequency.
5. Interpretability.
6. Privacy.
7. Resources.

Comparison Table

Aspect	Instance-Based	Model-Based
Training cost	Low	High
Prediction cost	High	Low
Memory	$O(nd)$	$O(\|\theta\|)$
Generalization	Local	Global
Updates	Easy (append)	Retrain needed
High-d handling	Poor	Better
Interpretability	Local examples	Varies
Privacy	Raw data stored	Parameters only

Pitfalls

- Ignoring feature scaling in IBL.
- High dimensionality issues.
- Class imbalance.
- Sensitivity to noise.

Pseudocode for kNN

```
Input: query x, training {(x_i,y_i)}_{i=1}^n, k, distance d
Compute distances: dist_i = d(x, x_i) for i=1..n
Find indices of k smallest dist_i -> neighbors N_k
Return majority vote (classification) or weighted average (regression)
```

Final Recommendations

- Small dataset and interpretability needed: use k-NN.
- Large dataset or high-dimensional: use model-based.
- For both power and local reasoning: embeddings + kNN.