

UNIT 1

Introduction to Machine Learning

(10 Hours) — Machine Learning (CACS456) · BCA IV/VIII

⚠ Please Note

These notes are prepared for quick study and **may contain errors or omissions**. Please cross-check formulas, definitions and worked examples with your textbook, class notes and teacher before relying on them in exams.

1.1 History of Machine Learning

Machine Learning (ML) is the field of study that gives computers the ability to learn from data without being explicitly programmed (Arthur Samuel, 1959). Its development has moved through several waves of optimism and disappointment.

Period	Milestone
1943	McCulloch & Pitts propose the first mathematical model of an artificial neuron.
1950	Alan Turing publishes “Computing Machinery and Intelligence” and proposes the Turing Test.
1952–59	Arthur Samuel writes a self-improving checkers program and coins the term “machine learning”.
1957–58	Frank Rosenblatt invents the Perceptron.
1969	Minsky & Papert show single-layer perceptrons cannot solve XOR → first “AI winter”.
1986	Rumelhart, Hinton & Williams popularise backpropagation for multi-layer networks.
1990s	Support Vector Machines, decision-tree methods, and statistical learning theory mature.
1997	IBM Deep Blue defeats world chess champion Garry Kasparov.
2006–12	Deep learning revives; ImageNet 2012 won by AlexNet (CNN).
2016 onward	AlphaGo, Transformers (2017), large language models and generative AI.

1.2 Brain–Neuron Learning System

Artificial neural networks are inspired by the biological brain, which contains roughly 10^{11} neurons, each connected to thousands of others through synapses. A biological neuron receives signals through **dendrites**, sums them in the **cell body (soma)**, and if the total exceeds a threshold it fires a signal along the **axon** to other neurons. Learning in the brain corresponds to strengthening or weakening synaptic connections.

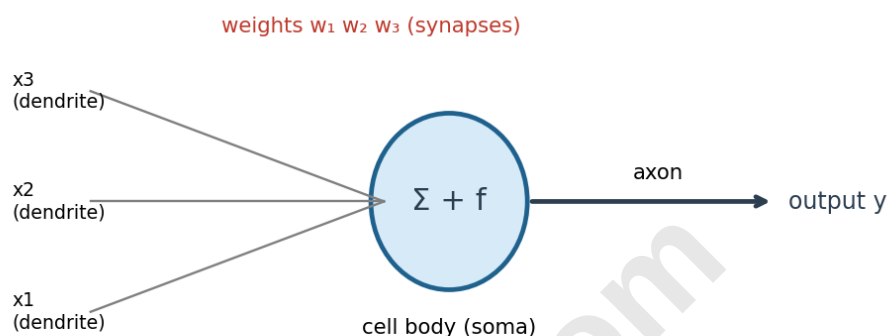


Figure 1.1: A neuron as a computational unit (inputs, weights, summation, activation, output).

An artificial neuron mimics this: $y = f(w_1x_1 + w_2x_2 + \dots + w_nx_n + b)$, where w_i are weights (synapse strengths), b is the bias and f is an activation function such as step, sigmoid or ReLU.

Biological	Artificial
Dendrites	Inputs x_i
Synapse strength	Weights w_i
Cell body	Summation Σ
Firing threshold	Activation function f / bias b
Axon	Output y

1.3 Definition and Types of Learning

Formal Definition (Tom Mitchell, 1997)

A computer program is said to learn from experience **E** with respect to some task **T** and performance measure **P**, if its performance at **T**, as measured by **P**, improves with experience **E**.

Example: **T** = classifying emails as spam; **E** = a set of labelled emails; **P** = percentage classified correctly.

Types of Learning

Type	Data	Goal	Examples
Supervised	Labelled (x, y)	Learn mapping $x \rightarrow y$	Spam detection, house-price prediction
Unsupervised	Unlabelled x	Discover structure	Customer clustering, PCA
Semi-supervised	Few labelled + many unlabelled	Use unlabelled data to improve a supervised model	Web-page classification, medical images
Reinforcement	Rewards from environment	Learn policy maximising cumulative reward	Game playing, robotics

- **Classification** – output is a discrete class (spam / not spam).
- **Regression** – output is a continuous number (price, temperature).
- **Clustering** – group similar items without labels.
- **Dimensionality reduction** – compress features while keeping information.

1.4 Need for Machine Learning

- Problems too complex to write explicit rules for (speech, vision, language).
- Tasks where rules change over time (fraud detection, recommendations).
- Huge data volumes that humans cannot analyse manually.
- Personalisation at scale (search, e-commerce, streaming).
- Discovering hidden patterns and insights (healthcare, genomics).
- Automation of repetitive decisions, lowering cost and error.

1.5 Data and Tools

Data is the fuel of ML. A dataset is a collection of **instances (samples)**, each described by **features (attributes)** and, in supervised learning, a **label (target)**. Data types include numerical, categorical, ordinal, text, image and time series. Good ML needs quality data: free of missing values, outliers, duplicates and bias, which is addressed by **pre-processing** (cleaning, normalisation, encoding, feature selection).

Tool	Use
Python	Most popular ML programming language

Tool	Use
NumPy / Pandas	Numerical arrays and data-frame manipulation
Matplotlib / Seaborn	Visualisation
Scikit-learn	Classical ML algorithms (regression, SVM, trees, k-NN, clustering, PCA)
TensorFlow / Keras / PyTorch	Neural networks and deep learning
Jupyter Notebook / Google Colab	Interactive development environment
R, Weka, MATLAB	Alternative statistical / GUI tools

1.6 Review of Statistics

Concept	Formula / Meaning
Mean (μ)	$\mu = (1/N) \sum x_i$ — average value
Median / Mode	Middle value / most frequent value
Variance (σ^2)	$\sigma^2 = (1/N) \sum (x_i - \mu)^2$ — spread of data
Standard deviation	$\sigma = \sqrt{\sigma^2}$
Covariance	$\text{cov}(X, Y) = (1/N) \sum (x_i - \mu_x)(y_i - \mu_y)$
Correlation (ρ)	$\rho = \text{cov}(X, Y) / (\sigma_x \sigma_y)$, range -1 to $+1$
Probability $P(A)$	$0 \leq P(A) \leq 1$; $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
Normal distribution	Bell-shaped curve defined by μ and σ

Example 1.1 — Mean and Variance

Data: 2, 4, 4, 4, 5, 5, 7, 9. Find mean, variance and standard deviation.

Solution

Mean $\mu = (2+4+4+4+5+5+7+9)/8 = 40/8 = 5$

Squared deviations: 9, 1, 1, 1, 0, 0, 4, 16 $\rightarrow$ sum = 32

Variance $\sigma^2 = 32/8 = 4$; Standard deviation $\sigma = 2$

1.7 Training, Validation and Test Data



Figure 1.2: Typical split of a dataset (ratios vary, e.g. 70/15/15 or 80/20).

- **Training set** – used to learn model parameters (weights).
- **Validation set** – used to tune hyper-parameters (e.g. k in k -NN, tree depth) and select models.
- **Test set** – used once at the end to estimate performance on unseen data; must never influence training.
- **Cross-validation (k-fold)** – data is split into k folds; each fold serves once as validation set and the scores are averaged. Useful for small datasets.

1.8 Theory of Learning

Feasibility of Learning

The learning problem: there is an unknown target function $f: X \rightarrow Y$. We have training examples $(x_1, y_1), \dots, (x_n, y_n)$ and a hypothesis set H . A learning algorithm picks a hypothesis $g \in H$ that approximates f . The question is whether what we see in-sample (training data) tells us anything about out-of-sample data. Without assumptions, learning is impossible, since f could take any value on unseen points. Feasibility comes from **probability**: if the data are drawn independently from a fixed distribution, then the in-sample error E_{in} is close to the out-of-sample error E_{out} with high probability.

Using Hoeffding's inequality for a single fixed hypothesis h :

Hoeffding Inequality

$$P(|E_{\text{in}}(h) - E_{\text{out}}(h)| > \epsilon) \leq 2 \cdot e^{-2\epsilon^2 N}$$

Larger N makes the probability of a big gap exponentially small.

When we choose g from a finite hypothesis set with M hypotheses, the union bound gives $P(|E_{\text{in}}(g) - E_{\text{out}}(g)| > \epsilon) \leq 2M \cdot e^{-2\epsilon^2 N}$. Larger M (more complex H) weakens the guarantee.

Error and Noise

- **In-sample error E_{in}** – error on training data. **Out-of-sample error E_{out}** – expected error on new data (the true goal).
- **Error measures:** squared error (regression) $(h(x) - f(x))^2$; 0/1 error (classification) $[h(x) \neq f(x)]$.
- **Noise** – real data is rarely a clean function of x . We model $y \sim P(y | x)$, i.e. a target distribution = deterministic part $f(x)$ + noise (measurement errors, missing features, labelling mistakes). Noise limits how low E_{out} can go.

Training versus Testing

Training uses the data to pick g ; testing verifies g on fresh data. The hope is $E_{out} \approx E_{in} \approx 0$. Two separate requirements exist:

- $E_{out}(g) \approx E_{in}(g)$ — a **generalisation** condition (depends on N and complexity of H).
- $E_{in}(g) \approx 0$ — an **approximation** condition (depends on how rich H is and how well it is optimised).

For infinite hypothesis sets the factor M is replaced by the **growth function** $m_h(N)$ (number of distinct dichotomies H can produce on N points), which is bounded by a polynomial if H has a finite **VC dimension dVC**. The VC dimension is the largest N that H can shatter (e.g. a 2-D perceptron has $dVC = 3$).

Generalization Bound

VC Generalization Bound

With probability $\geq 1 - \delta$: $E_{out}(g) \leq E_{in}(g) + \Omega(N, H, \delta)$

where $\Omega = \sqrt{(8/N) \cdot \ln(4 \cdot m_h(2N) / \delta)}$ is the *generalisation penalty*.

Meaning: penalty grows with model complexity (dVC) and shrinks with more data N . Rule of thumb: $N \geq 10 \cdot dVC$.

Approximation–Generalization Tradeoff

We want a hypothesis set rich enough to approximate f well (small E_{in}), yet simple enough to generalise (small penalty). A complex H gives better approximation but worse generalisation; a simple H gives the opposite. The best choice matches the model complexity to the *data resources*, not to the complexity of the unknown target.

1.9 Bias and Variance

The expected squared error of a learned model can be decomposed (for a fixed point x) as:

Bias–Variance Decomposition

$$E[(g(x) - f(x))^2] = \text{Bias}^2 + \text{Variance} + \text{Irreducible noise}$$

Bias = how far the average prediction (over many training sets) is from the true f . Caused by overly simple assumptions → underfitting.

Variance = how much predictions vary across different training sets. Caused by overly flexible models → overfitting.

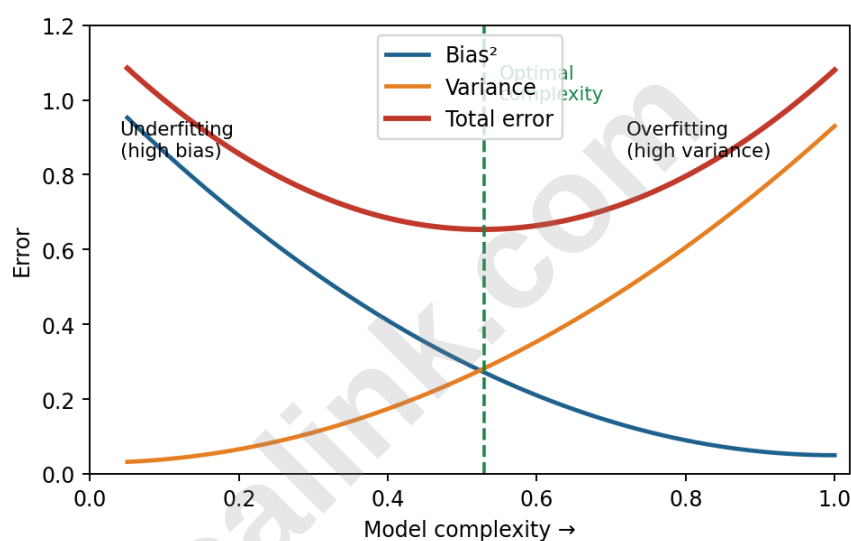


Figure 1.3: Bias–variance tradeoff — total error is minimised at moderate model complexity.

	High Bias	High Variance
Symptom	Poor on train AND test	Great on train, poor on test
Model	Too simple (e.g. line for curved data)	Too complex (e.g. degree-15 polynomial)
Fix	More features, complex model, less regularisation	More data, simpler model, regularisation, bagging

1.10 Learning Curves

A learning curve plots training error and validation error against the training set size N (or against training iterations). It diagnoses model behaviour:

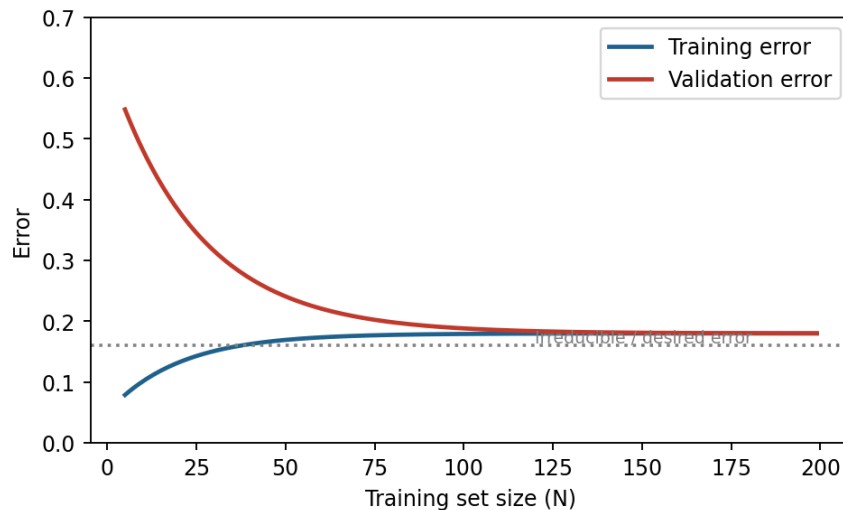


Figure 1.4: Learning curve — with more data the validation error falls and training error rises until they converge.

- **High bias (underfitting):** both curves converge quickly to a *high* error; adding more data does not help — use a more complex model.
- **High variance (overfitting):** a large gap between low training error and high validation error; more data or regularisation helps.
- **Good fit:** both curves converge to a low error with a small gap.

Key Takeaways — Unit 1

- ✓ ML = learning from experience E for task T measured by P (Mitchell).
- ✓ Types: supervised, unsupervised, semi-supervised, reinforcement.
- ✓ Split data into training / validation / test; never tune on the test set.
- ✓ Hoeffding: E_{in} tracks E_{out} as N grows; complexity of H (VC dimension) controls the gap.
- ✓ Bias = underfitting, variance = overfitting; aim for the sweet spot.
- ✓ Learning curves reveal whether you need more data or a different model.

— End of Unit 1 —

These notes may contain errors — please verify with your textbook and teacher.