

UNIT 3

Machine Learning

(12 Hours) — Knowledge Engineering (CACS458)

⚠ Please Note

These notes are generated to help with revision and **may contain errors or omissions**. Please verify important definitions, formulas and examples against your textbook, lecture notes and the official TU syllabus before relying on them in an exam.

3.1 Machine Learning and Its Applications

Machine learning (ML) is a branch of AI in which computer programs improve their performance on a task automatically through experience (data), without being explicitly programmed with every rule. Tom Mitchell's definition: *a program learns from experience E with respect to task T and performance measure P if its performance on T , as measured by P , improves with E .*

In knowledge engineering, ML helps overcome the **knowledge acquisition bottleneck** by learning rules and patterns directly from data.

Typical ML Workflow

- Collect data → clean and pre-process → select features.
- Split data into **training**, **validation** and **test** sets.
- Train a model → evaluate on unseen data → tune → deploy and monitor.
- **Overfitting**: model memorises training data and generalises poorly. **Underfitting**: model is too simple to capture the pattern.

Applications of Machine Learning

Area	Application
Healthcare	Disease prediction, medical image analysis
Finance	Credit scoring, fraud detection, stock trend analysis
E-commerce / media	Recommendation systems (products, movies)
Language	Spam filtering, translation, sentiment analysis, chatbots

Area	Application
Vision	Face recognition, object detection, OCR
Transport	Self-driving vehicles, route prediction
Agriculture	Crop yield prediction, plant-disease detection

3.2 Supervised and Unsupervised Learning

Basis	Supervised	Unsupervised	Reinforcement
Data	Labelled (input + known output)	Unlabelled	Feedback as reward/penalty
Goal	Learn mapping from input to output	Discover hidden structure/groups	Learn best actions to maximise reward
Tasks	Classification, regression	Clustering, association, dimensionality reduction	Control, games, robotics
Examples	Spam detection, house price prediction	Customer segmentation	Game-playing agent

Semi-supervised learning uses a small amount of labelled data with a large amount of unlabelled data.

Note: Regression predicts a continuous value (price, temperature); classification predicts a discrete label (spam / not spam).

3.3 Classification and Clustering

Classification

Classification assigns an input object to one of a predefined set of classes using a model learned from labelled examples. Binary classification has two classes; multi-class has more than two.

Evaluating a Classifier

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

Accuracy = $(TP + TN) / (TP + TN + FP + FN)$; **Precision** = $TP / (TP + FP)$; **Recall** = $TP / (TP + FN)$; **F1** = $2PR / (P + R)$.

Example 3.1 — Confusion matrix

A spam filter gives TP = 40, FP = 10, FN = 5, TN = 45. Compute accuracy, precision and recall.

Solution

Accuracy = $(40 + 45) / 100 = 0.85$

Precision = $40 / (40 + 10) = 0.80$

Recall = $40 / (40 + 5) = 0.889$

Clustering

Clustering groups unlabelled objects so that objects in the same cluster are more similar to each other than to those in other clusters.

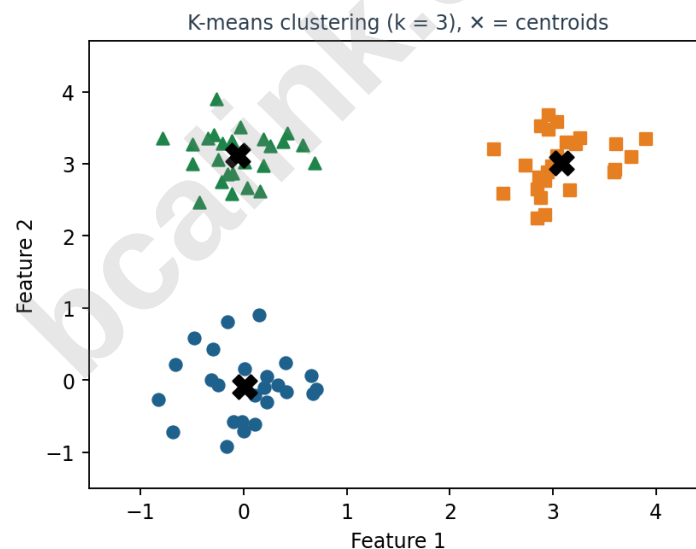


Figure 3.1: K-means clustering with three clusters and their centroids.

- **K-means**: choose k centroids → assign each point to nearest centroid → recompute centroids → repeat until stable.
- **Hierarchical clustering**: builds a tree (dendrogram) by merging (agglomerative) or splitting (divisive) clusters.
- **DBSCAN**: density-based; finds arbitrarily shaped clusters and noise.
- Similarity is measured by distance, e.g. Euclidean: $d(x, y) = \sqrt{\sum (x_i - y_i)^2}$.

Example 3.2 — One K-means step

Points: A(1,1), B(2,1), C(4,3), D(5,4). Initial centroids: $C_1 = (1,1)$, $C_2 = (5,4)$. Perform one iteration.

Solution

Assign: $A \rightarrow C_1$ (distance 0); $B \rightarrow C_1$ (1 vs 4.24); $C \rightarrow C_2$ (3.6 vs 1.41); $D \rightarrow C_2$ (0).

Clusters: {A, B} and {C, D}.

New centroids: $C_1 = ((1+2)/2, (1+1)/2) = (1.5, 1)$; $C_2 = ((4+5)/2, (3+4)/2) = (4.5, 3.5)$.

3.4 Classification Algorithms

3.4.1 Linear Classifiers

A linear classifier decides the class from a linear combination of features: $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + \mathbf{b}$. If $f(\mathbf{x}) \geq 0$ predict class +1, else class -1. The decision boundary is a line (2-D) or hyperplane (higher dimensions). Examples: **perceptron**, **logistic regression**, linear discriminant analysis, linear SVM.

- **Perceptron update rule:** if a point is misclassified, $\mathbf{w} \leftarrow \mathbf{w} + \eta \cdot \mathbf{y} \cdot \mathbf{x}$ and $\mathbf{b} \leftarrow \mathbf{b} + \eta \cdot \mathbf{y}$ (η = learning rate).
- **Logistic regression** passes the linear score through the sigmoid $\sigma(z) = 1 / (1 + e^{-z})$ to obtain a probability.
- Works only when classes are (nearly) *linearly separable*; fast and easy to interpret.

3.4.2 Nearest Neighbour (k-NN)

k-nearest neighbour is a lazy, instance-based learner: it stores all training data and classifies a new point by the majority class among its k closest training points (distance usually Euclidean).

- Small $k \rightarrow$ sensitive to noise (overfitting); large $k \rightarrow$ smoother but may blur classes.
- Features should be scaled; slow for large datasets; no explicit training phase.

Example 3.3 — k-NN

Training: P1(1,1)=Red, P2(2,2)=Red, P3(6,5)=Blue, P4(7,7)=Blue, P5(5,6)=Blue. Classify X(3,2) with $k = 3$.

Solution

$d(X,P1) = \sqrt{(4+1)} = 2.24$; $d(X,P2) = \sqrt{(1+0)} = 1.0$; $d(X,P3) = \sqrt{(9+9)} = 4.24$; $d(X,P4) = \sqrt{(16+25)} = 6.40$; $d(X,P5) = \sqrt{(4+16)} = 4.47$.

3 nearest: P2 (Red), P1 (Red), P3 (Blue) $\rightarrow$ majority = **Red**.

3.4.3 Support Vector Machines (SVM)

An SVM finds the hyperplane that separates two classes with the **maximum margin** — the largest distance to the nearest training points, called **support vectors**. Maximising the margin $2/\|w\|$ is equivalent to minimising $\frac{1}{2}\|w\|^2$ subject to $y_i(w \cdot x_i + b) \geq 1$.

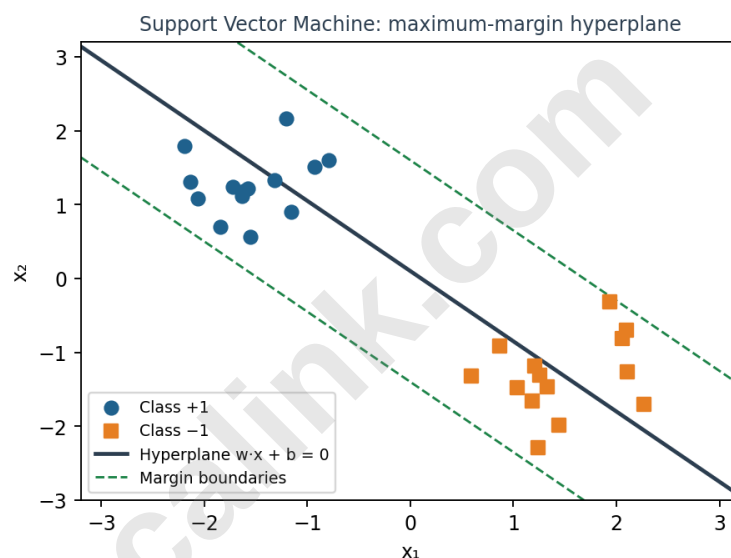


Figure 3.2: SVM with maximum-margin hyperplane (dashed lines mark the margin).

- **Soft margin:** parameter C allows some misclassification for noisy data.
- **Kernel trick:** maps data implicitly to higher dimensions to separate non-linear data. Common kernels: linear, polynomial, RBF (Gaussian).
- Effective in high-dimensional spaces; less suited to very large datasets; needs scaling and parameter tuning.

3.4.4 Decision Tree

A decision tree classifies by asking a series of questions on attributes. Internal nodes test an attribute, branches are outcomes, and leaves give the class. The ID3 algorithm chooses the attribute with the highest **information gain**.

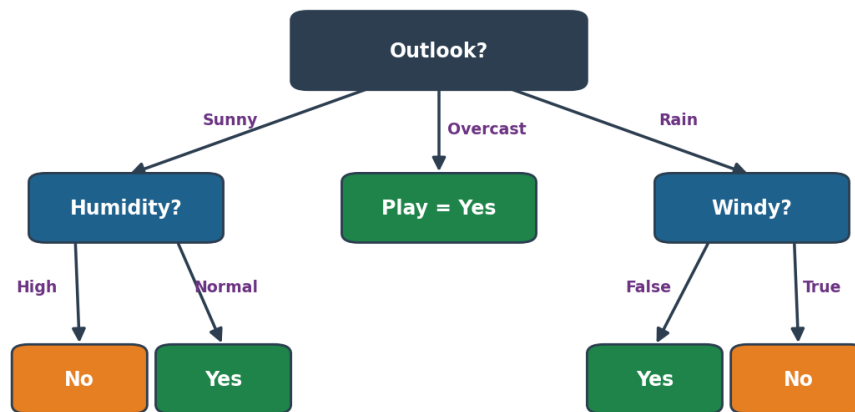


Figure 3.3: A decision tree for the “play tennis” problem.

Entropy: $H(S) = - \sum p_i \log_2 p_i$. **Information gain:** $\text{Gain}(S, A) = H(S) - \sum (|S_v| / |S|) \cdot H(S_v)$.
 Alternatives: C4.5 (gain ratio) and CART (Gini index = $1 - \sum p_i^2$).

Example 3.4 — Entropy

A set S has 9 positive and 5 negative examples. Find $H(S)$.

Solution

$p_+ = 9/14 = 0.643$; $p_- = 5/14 = 0.357$.

$H(S) = -0.643 \cdot \log_2(0.643) - 0.357 \cdot \log_2(0.357) = 0.643 \times 0.637 + 0.357 \times 1.485 \approx 0.410 + 0.530 = 0.940$.

Note: Trees are easy to interpret and convert directly to IF-THEN rules (useful in knowledge bases), but can overfit; pruning reduces this.

3.4.5 Random Forest

A **random forest** is an *ensemble* of many decision trees. Each tree is trained on a random bootstrap sample of the data (**bagging**) and considers only a random subset of features at each split. Final prediction is by **majority vote** (classification) or average (regression).

- Reduces overfitting and variance compared with a single tree.
- Handles large feature sets, missing values and gives feature-importance scores.
- Less interpretable than one tree; slower to train and predict.

3.4.6 Neural Networks

An **artificial neural network (ANN)** is inspired by the brain. It consists of layers of connected neurons. A neuron computes $y = \varphi(\sum w_i x_i + b)$, where φ is an activation function (sigmoid, tanh, ReLU).

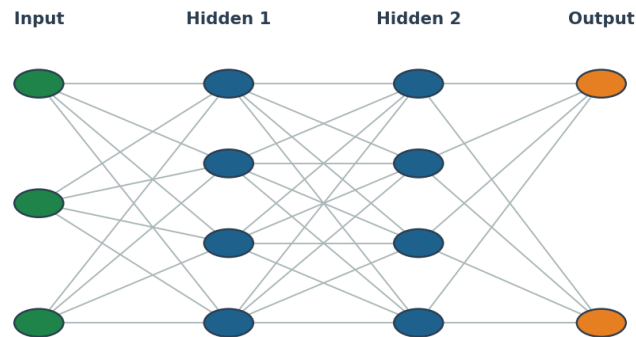


Figure 3.4: A feed-forward neural network with two hidden layers.

- **Input layer** receives features; **hidden layers** learn intermediate representations; **output layer** gives predictions.
- **Training by backpropagation:** forward pass computes output and error; backward pass propagates the error and adjusts weights by gradient descent: $w \leftarrow w - \eta \partial E / \partial w$.
- Deep networks (many hidden layers) form **deep learning**: CNNs for images, RNNs/Transformers for sequences.
- Powerful and flexible but need much data and computing power, and behave as ‘black boxes’.

Example 3.5 — Single neuron

Inputs $x_1 = 1$, $x_2 = 0$; weights $w_1 = 0.5$, $w_2 = -0.3$; bias $b = 0.1$; step activation (output 1 if sum ≥ 0). Find the output.

Solution

Sum = $0.5 \times 1 + (-0.3) \times 0 + 0.1 = 0.6 \geq 0 \rightarrow \text{output} = 1$.

3.4.7 Case-Based Reasoning (CBR)

Case-based reasoning solves new problems by reusing solutions of similar past problems (cases) stored in a case base — much as a doctor recalls similar earlier patients. A case contains a *problem description*, a *solution* and often the *outcome*.

The CBR cycle (4 Rs):

- **Retrieve** the most similar case(s) from the case base.
- **Reuse** the retrieved solution for the new problem.
- **Revise** (adapt) the solution if it does not fit exactly and test it.
- **Retain** the new, confirmed case in the case base for future learning.
- Advantages: no need for explicit rules; learns incrementally; natural for experts.
- Limitations: needs a large and well-indexed case base; similarity measure and adaptation are hard to design.

Comparison of Algorithms

Algorithm	Type	Strength	Weakness
Linear classifier	Parametric	Fast, simple	Only linear boundaries
k-NN	Instance-based	No training, simple	Slow on large data
SVM	Margin-based	Good in high dimensions	Tuning, kernel choice
Decision tree	Rule-based	Interpretable	Overfits
Random forest	Ensemble	Accurate, robust	Less interpretable
Neural network	Layered	Handles complex patterns	Data-hungry, black box
CBR	Memory-based	Reuses experience	Needs case base maintenance

Key Takeaways — Unit 3

- ✓ Machine learning improves performance through experience; supervised learning uses labelled data, unsupervised learning finds structure in unlabelled data.
- ✓ Classification predicts discrete classes; clustering (e.g. K-means) groups similar unlabelled objects.
- ✓ Evaluate with accuracy, precision, recall and F1 from the confusion matrix.
- ✓ k-NN uses nearest examples; SVM maximises margin; decision trees use information gain; random forests combine many trees.
- ✓ Neural networks learn through backpropagation; CBR solves new problems from past cases (retrieve, reuse, revise, retain).

— End of Unit 3 —