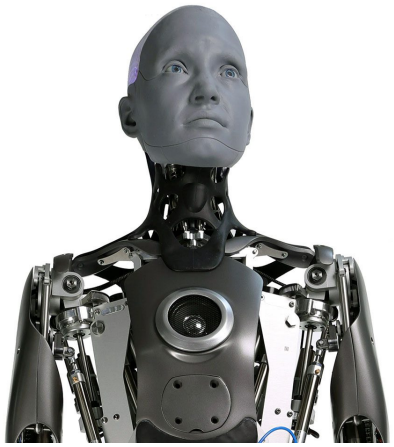
Artificial Intelligence (AI)

A-Level Computer Science

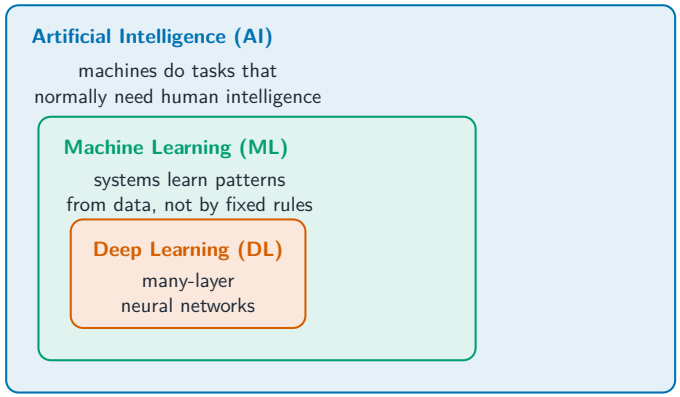
What AI is

Artificial intelligence 人工智能 (AI) builds systems that do tasks normally needing human intelligence — recognising speech and images, translating, playing games, driving, generating text. Most modern AI uses **machine learning** 机器学习 — algorithms that learn patterns from data instead of being programmed step by step. Within it, **deep learning** 深度学习, using **neural networks** 神经网络 with many layers, has been dominant since the 2010s.

A **humanoid robot** 人形机器人 puts many of these abilities into one body: it uses AI to see faces, understand speech and move its face and arms in a lifelike way.



A humanoid robot uses AI to see, listen and respond like a person

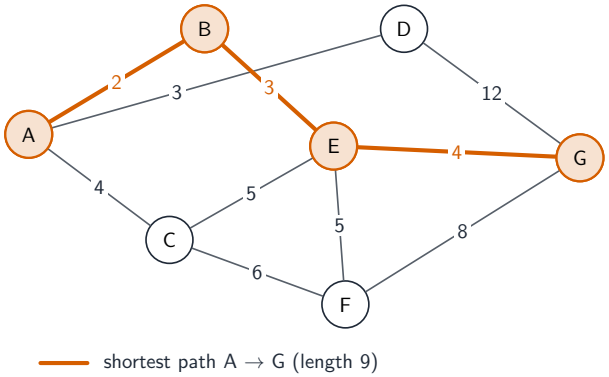


Deep learning is part of machine learning, which is part of AI

Graphs in AI

Many AI problems sit on a **graph** 图 — **nodes** 节点 (states, places) joined by **edges** 边 (moves, relationships).

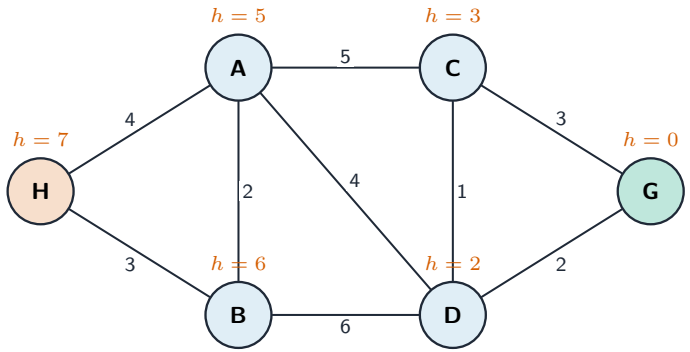
- **pathfinding**: roads form a graph; the shortest route is a graph search (**Dijkstra's algorithm**, the **A* algorithm**).
- **game playing**: each board position is a node, each move an edge; **minimax** 极小化极大 with alpha-beta pruning searches the game tree.
- **state-space search**: a planning problem is moving between states by applying operators to reach a goal.
- **knowledge representation**: a **semantic network** 语义网络 has concepts as nodes and relationships as edges ("dog IS-A animal"); a **knowledge graph** 知识图谱 stores facts about the world for search engines and assistants.



AI problems often sit on a graph; here the shortest path is highlighted

Standard tools for navigating graphs include **breadth-first search** 广度优先搜索 and **depth-first search** 深度优先搜索.

"Describe the purpose and structure of a graph in an AI system." *Purpose:* to **represent a problem** as a set of states (or places) and the possible moves between them, so that an **algorithm can search** it for a solution, such as the shortest or cheapest route, or the best next move. *Structure:* a set of **nodes** (vertices), each representing a state, location or item, joined by **edges** representing the connections between them; each edge may carry a **weight** (a cost, distance or time), and edges may be **directed** (one-way) or undirected. **"Explain the use of graphs to aid AI":** the graph is the model on which the AI's search algorithms run: A* and Dijkstra's algorithm find optimal paths through it (navigation, routing), game positions form a tree searched for the best move, and knowledge stored as a graph lets a system reason about how facts are related.



edge numbers: actual distances; h (red): the heuristic estimate of the distance still to go to G, used only by A*. Start at H (Home), goal G (School)

The graph used below: the edge numbers are real distances; the red numbers are each node's heuristic 启发式 *estimate of how far the goal still is, which only A* uses*

Dijkstra's algorithm. It finds the **shortest distance from the start to every node**. Keep a table of the best distance found so far to each node (start 0, all others infinity). Repeatedly take the unvisited node with the **smallest** distance, mark it visited, and for each neighbour check whether going through this node gives a shorter distance; if so, update it and record where it came from. Stop when every node is visited (or the target is).

Worked example. Find the shortest distances from H to every other node in the graph above.

step	visit	H	A	B	C	D	G
start		0	∞	∞	∞	∞	∞
1	H (0)	0	4 (H)	3 (H)	∞	∞	∞
2	B (3)	0	4 (H)	3	∞	9 (B)	∞
3	A (4)	0	4	3	9 (A)	8 (A)	∞
4	D (8)	0	4	3	9 (A)	8	10 (D)
5	C (9)	0	4	3	9	8	10 (D)
6	G (10)						

Shortest distances: A 4, B 3, D 8, C 9, G 10, and the path to G is H–A–D–G (read the "came from" labels backwards). At step 3, A offers D a distance of $4 + 4 = 8$, better than the 9 found through B, so D is updated; at step 5, C could reach G at $9 + 3 = 12$, worse than 10, so nothing changes. Showing these comparisons is the "working" the question asks for.

The A* algorithm. Dijkstra explores in every direction. A* adds a **heuristic** h, an estimate of the distance still to go, and always expands the node with the smallest $f = g + h$, where g is the distance travelled so far. With a sensible heuristic (never over-estimating), it finds the same shortest path while looking at far fewer nodes, which is why satnavs and games use it. The exam gives h for each node and a table to fill in.

Worked example. Find a path from H to G with A*, showing the working.

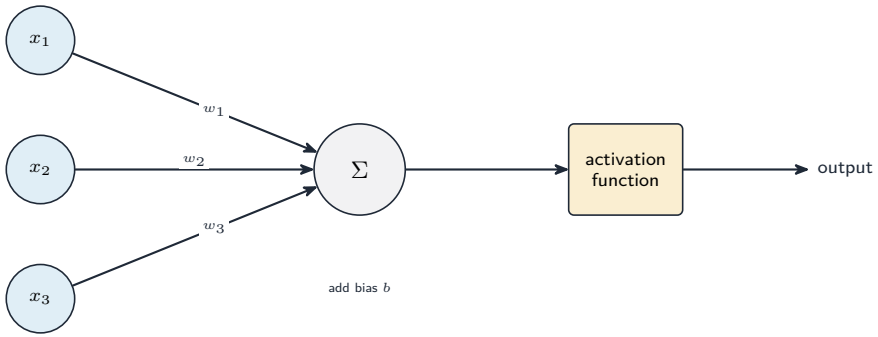
node expanded	g so far	h	$f = g+h$	neighbours added (node: g, h, f)
H	0	7	7	A: 4, 5, 9; B: 3, 6, 9
B (tie with A; either)	3	6	9	D via B: 9, 2, 11
A	4	5	9	C: 9, 3, 12; D via A: 8, 2, 10 (better than 11, keep)
D	8	2	10	G: 10, 0, 10; C via D: 9 (no better)
G	10	0	10	goal reached

Path H–A–D–G, length 10, the same as Dijkstra's, but C was never expanded. Each time a node is reached by a second route, keep the smaller g; the search ends when the **goal** is the node with the smallest f. State the g, h and f values in every row: those are the marks.

Artificial neural networks (ANNs)

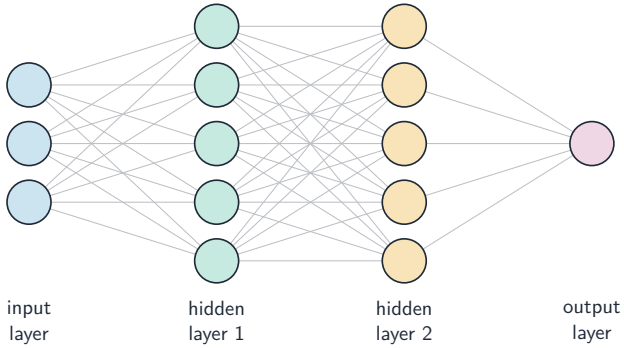
An ANN is inspired by the brain's neurons. An **artificial neuron** 人工神经元:

- takes several **input values**, multiplies each by a **weight** 权重, and adds them up with a **bias term** 偏置项.
- applies an **activation function** 激活函数 (a non-linear function such as ReLU) to the sum.
- outputs the result, which feeds neurons further on.



A single neuron: each input times its weight, summed with a bias, then an activation function

Neurons sit in layers: an **input layer**, one or more **hidden layers** 隐藏层 (where useful internal patterns are learned), and an **output layer**. With many hidden layers it is a **deep neural network** 深度神经网络, and training it is deep learning.



A neural network with an input layer, two hidden layers and an output layer

ANNs let models learn complex patterns straight from raw data (pixels, audio, text) without hand-designed features — driving breakthroughs in **image recognition** 图像识别, **speech recognition** 语音识别, **machine translation** 机器翻译, and game playing. They do well with large amounts of data, noisy or very complex input, and patterns too hard to capture with explicit rules.

"Explain what is meant by an artificial neural network." *A model of the brain's network of neurons, made of layers of connected nodes: an input layer, one or more hidden layers and an output layer.* Each connection has a **weight**; each node sums

its weighted inputs and passes the result through an activation function to the next layer. **"Explain how ANNs enable machine learning" (three marks):** the network is **trained** on many examples; for each example the output is compared with the expected result and the **error** is used to **adjust the weights** (back propagation) so that the error falls; after enough examples the weights encode the **patterns** in the data, and the network can then classify or predict for **new** data it has never seen. **"State the reason for multiple hidden layers":** each additional layer combines the features found by the layer before it into **more complex, more abstract** features, so the network can learn **more complex** relationships (edges, then shapes, then objects); that is what makes a network deep.

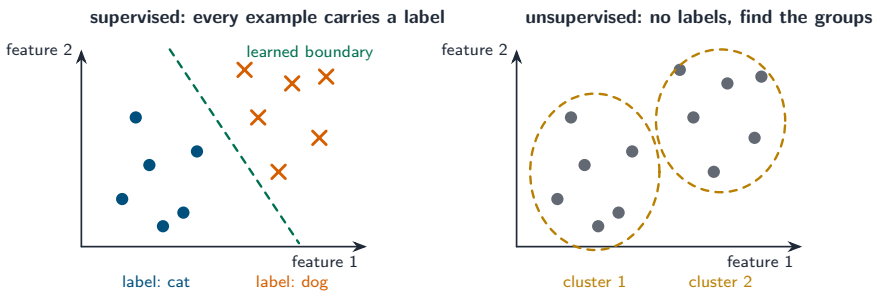
Machine learning, deep learning, reinforcement learning

Machine learning

The umbrella term — any algorithm that learns from data. Three paradigms:

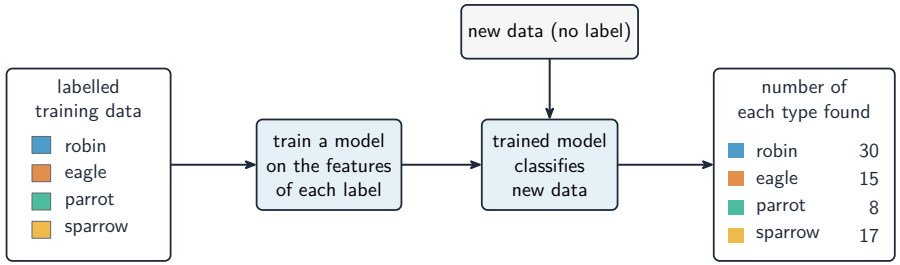
- **supervised learning** 监督学习 — the data has **labels** 标签 (images tagged "cat"/"dog"); the algorithm learns input → label. Used for **classification** 分类 (a category) and regression.
- **unsupervised learning** 无监督学习 — no labels; the algorithm finds structure, e.g. a **cluster** 聚类 of similar customers.
- reinforcement learning (below).

Use ML when explicit rules would be impractical (spam filters, recommendations, fraud detection).



The same data seen two ways: with labels, the task is to learn what separates the classes; without labels, the task is to discover that there are groups at all

"Describe supervised learning and unsupervised learning" (the marked wordings). *Supervised learning:* the algorithm is trained on **labelled training data** 训练数据, each example paired with the **correct output** (the target); it learns the **relationship between inputs and outputs** and uses it to classify or predict for new inputs; the answers are known while training, so the error can be measured. *Unsupervised learning:* the data is **unlabelled**, with no correct answers given; the algorithm looks for **patterns, structure or groupings** in the data by itself (clustering similar items, finding associations); the output is a set of categories or relationships that were not defined in advance. **How they differ:** labelled against unlabelled data; known outputs against discovered structure; supervised is used to **predict** (classification, regression), unsupervised to **explore** (clustering, anomaly detection). Both are categories of machine learning; the third is reinforcement learning.



Supervised learning: a model is trained on labelled data, then recognises new data

Deep learning

A subset of ML using deep neural networks. Lower layers learn simple patterns (edges, phonemes), higher layers combine them into abstract concepts. It needs **lots of data** and **lots of compute** (GPUs); for small datasets, simpler ML methods often do better.

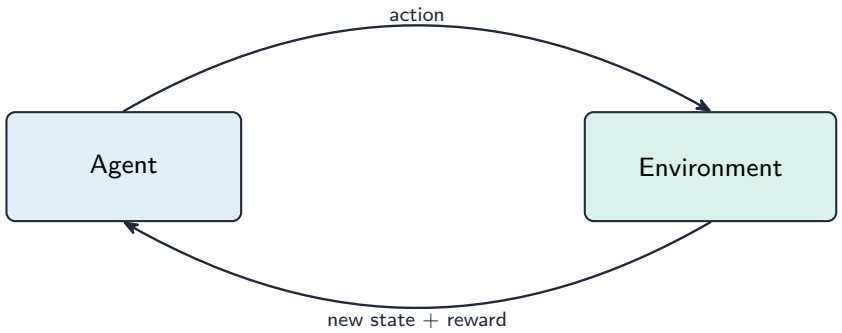
"Explain what is meant by deep learning" (three marks). *Machine learning that uses artificial neural networks with **many hidden layers** (deep networks); the network is trained on **very large amounts of data**, and each layer **extracts features** from the output of the layer below, so that the network learns the features it needs by itself rather than having them specified by the programmer.* **Reasons for using it:** it can solve problems too **complex** for hand-written rules or shallow models (recognising faces, understanding speech, translating text); it **improves as more data** becomes available; it removes the need for human **feature engineering**; and it can handle **unstructured** data such as images, sound and text. **How it is made more effective:** more (and better-labelled) training data; more layers or nodes, within the limits of overfitting; more processing power (GPUs) and training time; tuning the learning rate and other parameters. **Examples:** speech recognition

in voice assistants, image recognition in medical scans and self-driving cars, machine translation, recommendation systems.

Reinforcement learning

In **reinforcement learning** 强化学习, an **agent** 智能体 acts in an environment; each action changes the state and returns a **reward** 奖励. The agent learns a **policy** 策略 (a strategy) that maximises the total reward over time, by **trial and error** with no labels up front. Used for sequential-decision problems — games, robot control, autonomous driving.

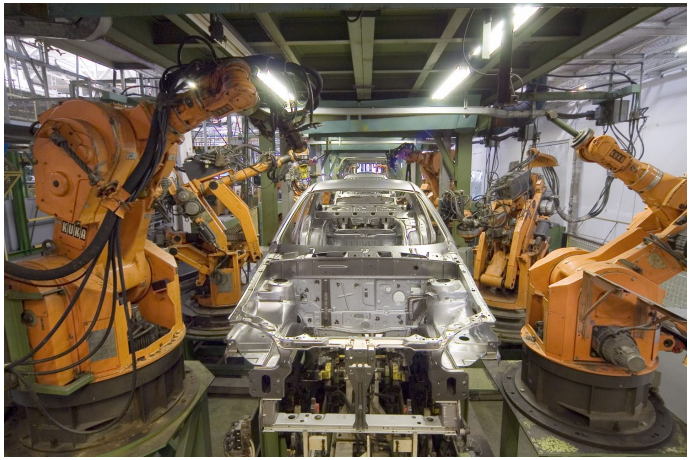
"Explain what is meant by reinforcement learning" (three marks). *An **agent** learns by **interacting with its environment**: it takes an action, the environment moves to a new state and returns a **reward** (or penalty), and the agent adjusts its behaviour so as to **maximise the total reward** over time.* There is **no labelled data**: the agent learns by **trial and error**, discovering which actions are good from the rewards it collects, and gradually forms a **policy** that says what to do in each state. Used where the right answer is not known in advance but the result of an action can be scored: game playing (chess, Go), robot control, traffic-light timing, resource allocation. A computer playing a board game against a user learns in this way, or searches the game tree with minimax to choose the move whose worst outcome is best.



Reinforcement learning: the agent acts, the environment returns a new state and a reward, and the agent learns from it

A **self-driving car** 自动驾驶汽车 is a real example. **Lidar** 激光雷达 and camera sensors (the spinning unit on the roof) build a live picture of the road, and a learned policy decides how to steer, speed up and brake safely.

A self-driving car uses cameras and lidar sensors to see the road around it



Industrial robot arms on a production line: reinforcement learning can teach a robot to control its movements

Training an ANN: backpropagation

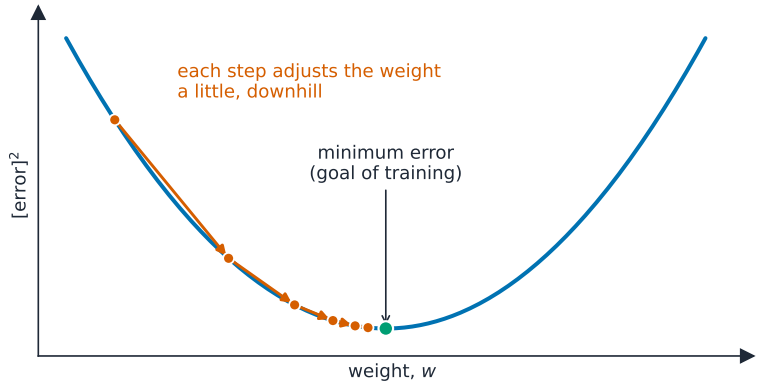
Training adjusts the **weights** so outputs match the targets. The standard method is **backpropagation** 反向传播 (back propagation of errors) with **gradient descent** 梯度下降. For each training example:

1. **forward pass** — feed the input through to the output.

2. **compute the error** with a **loss function** 损失函数 (a single number for how wrong the output is).
3. **backward pass** — propagate the error **backwards**, finding each weight's **gradient** (how much it contributed to the error) using the chain rule.
4. **update the weights** by a small step (set by the **learning rate** 学习率) that reduces the error.

Repeat over many examples and many passes (**epochs** 训练轮次) until the error stops shrinking. The name "back" comes from step 3: the error flows from the output **back** towards the input, so every weight's gradient is found in one sweep. After training, a new input needs only one forward pass to get a prediction.

"Describe the back propagation of errors method" (four marks). (1) An input is fed **forward** through the network and its output is compared with the **expected (target) output**; (2) the difference is the **error**; (3) the error is passed **backwards** through the network, layer by layer from the output to the input, and each weight's share of the error is calculated; (4) the **weights are adjusted** in proportion to their contribution, in the direction that **reduces** the error; (5) the process is **repeated** with many examples until the error is as small as required. The point of the method is that a network with hidden layers has no direct way of knowing which internal weight caused an output error; back propagation apportions the blame.



gradient descent: update w opposite the slope of the error surface

Training adjusts the weights to reach the minimum error

Regression

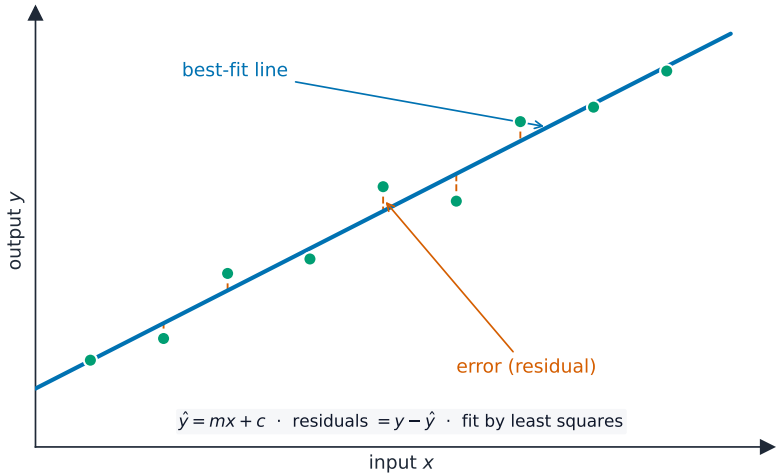
Some tasks predict a **number** (a house price, tomorrow's temperature) — **regression** 回归, as opposed to classification (a category).

Linear regression 线性回归 fits a straight line (or hyperplane):

$$y = m_1x_1 + m_2x_2 + \ldots + m_nx_n + c.$$

Choose the coefficients to **minimise the sum of squared errors** against the training data. Use it when the relationship looks roughly linear and you want an interpretable model. For curved data, use polynomial, decision-tree, or neural-network **regression methods** — same idea: define a model, define a loss, and adjust the parameters to minimise it. Regression and classification are both supervised; the choice depends on whether the answer is a number or a category.

"Describe regression methods in machine learning" (two marks). *Statistical methods that find the **relationship** between input variables and a **continuous** output, by fitting a function (a line or curve) to the training data with the smallest total error; the fitted function is then used to **predict** the output for new inputs.* Linear regression fits a straight line; other methods fit curves. Regression predicts a **value** (a price, a temperature, a time); classification predicts a **category**, which is the distinction the exam asks for.



Linear regression fits the line that makes the total squared error (the dashed gaps) as small as possible

How AI is used in a real scenario

Many exam scenarios use the same pattern — a **deep-learning model trained on labelled data**, often several combined into a pipeline:

- **customer identification** at an automated shop: the system is trained on labelled face images; a camera captures a face; image recognition extracts a representation; it is matched against registered customers; the closest match identifies the person.
- **reading text from images**: image recognition finds text regions; **optical character recognition** 光学字符识别 extracts the characters; machine translation converts them; **text-to-speech** 文本转语音 reads them aloud.
- **checkout item-detection**: object-detection AI, trained on labelled product images, sees which items go into a basket and charges the account.

By the time a user interacts with the system, the model is fast — it only does forward-pass inference; the intelligence is in the patterns learned during training.

Model answers for the scenario questions. *A car-park camera reads registration numbers:* the camera captures an image; an AI trained on many **labelled images** of number plates locates the plate in the image; **character recognition** (a deep-learning classifier, again trained on labelled characters) converts the plate into text; the text is stored with the time and matched when the car leaves. *A CCTV system detects and tracks a person:* image-recognition software trained on labelled images of people identifies a person in each frame; the system compares successive frames to follow their movement; unusual movement can trigger an alert. *Speech turned into commands:* speech recognition trained on many recorded voices converts the sound into text; the system matches the text to a set of known commands; it improves as it is corrected. *A camera that focuses on faces:* a face-detection model trained on labelled faces finds the face region, and the lens is adjusted to bring that region into focus. *A bank's face-recognition login:* the app captures the face, a deep network extracts its features, and they are compared with the stored features for that customer. In every case the pattern is: **trained on labelled examples, extracts features, matches or classifies new input**.

Worked example. For each task, say whether it needs **regression** or **classification**, and what the output layer of an ANN would look like: (a) predict tomorrow's temperature; (b) decide whether an email is spam. Ask what **kind of thing** is being predicted. (a) A temperature is a **number** on a continuous scale, so this is **regression**, and the output layer is a **single** neuron holding that value. (b) Spam or not-spam is a **category**, so this is **classification**, and the output gives a probability per class. Both are **supervised** learning: each needs labelled examples to train on, and training adjusts the **weights** by **backpropagation** to reduce the error. The deciding question is simply number-or-category - not how difficult the task feels.

Definitions the examiner accepts

A definition question is marked against fixed wording. Learn these exactly, and give one answer only.

Term	Definition
graph (in AI)	a set of nodes representing states or places, joined by edges representing connections, often weighted, that a search algorithm can explore
Dijkstra's algorithm	finds the shortest distance from a start node to every other node by always visiting the unvisited node with the smallest distance so far
A* algorithm	a shortest-path search that expands the node with the smallest total of distance so far plus a heuristic estimate of the distance to the goal
artificial neural network	a model of the brain's neurons: layers of nodes joined by weighted connections, trained by adjusting the weights
machine learning	algorithms that learn from data and improve with experience rather than following fixed rules
supervised learning	learning from labelled training data in which the correct output for each input is known
unsupervised learning	learning from unlabelled data by finding patterns, groupings or structure in it
reinforcement learning	an agent learns by trial and error, choosing actions in an environment to maximise the rewards it receives
deep learning	machine learning using neural networks with many hidden layers, trained on large amounts of data, each layer extracting features from the one below
back propagation of errors	comparing the network's output with the target, passing the error back through the layers and adjusting each weight to reduce it
regression	fitting a function to training data in order to predict a continuous output value from inputs

Exam tips

- Graph answers name nodes, edges and weights, and what they represent; then the algorithm. Dijkstra: table of distances, visit the smallest, update neighbours. A*: g, h and $f = g + h$ in every row, expand the smallest f.

- ANN answers name the layers, the weighted connections and training; deep learning adds many hidden layers, large data and automatic feature extraction, with a reason and an example.
- The three categories in one line each: labelled data and known outputs; unlabelled data and discovered structure; agent, environment, actions and rewards.
- Back propagation: compare with the target, error backwards through the layers, adjust weights to reduce it, repeat. Regression predicts a value; classification predicts a category.
- Scenario questions want the pipeline: trained on labelled examples, extracts features, recognises or classifies new input; name the type of AI (image recognition, speech recognition, deep learning).

Common mistakes

- Describing a graph as "a chart"; in AI it is nodes and edges.
- Running Dijkstra by picking the nearest neighbour of the current node rather than the smallest overall distance not yet visited; or forgetting to update a node when a shorter route appears.
- Adding h into g for the next step in A*; g is only the real distance, h is recomputed from the table.
- Saying deep learning is "learning a lot"; it is the many hidden layers.
- Confusing unsupervised learning with reinforcement learning; the first finds structure in data, the second learns from rewards.
- Describing back propagation without the comparison with the expected output or without saying the weights are adjusted.
- Calling a prediction of a price "classification"; a continuous value is regression.