Kilburn & Strode

LexisNexis

Artificial intelligence and machine learning—an introduction to the technology

Produced in partnership with Alexander Korenberg, partner, and Nathalie Richards, trainee patent attorney, of Kilburn & Strode LLP

This Practice Note explains the basics of artificial intelligence (AI) and machine learning (ML) technology.

It covers:

- The history of AI and ML
- The importance of data
- Training an ML model
- Types of ML
- Considerations when selecting or assessing an ML algorithm
- Neural networks
- What is deep learning?
- Common neural network architectures
- Some examples of other commonly used ML algorithms
- Key challenges for AI and ML—transparency, explainability and bias
- Privacy and data protection
- Protecting AI technology

This Practice Note does not consider legal and regulatory issues arising in connection with the use or development of AI or ML technologies.

The history of AI and ML

Although often thought of (and used) as a new exciting technology, AI has in fact been around for over 70 years. While sometimes assumed to be incomprehensible to anyone without a specialism in computing, AI is built on relatively simple mathematical concepts. Even today, when it has advanced into more computational complex algorithms, it is important to remember that it is just that, mathematical concepts implemented in software and written by a person.

As aptly put by Gary Smith in *The AI Delusion (OUP 2018) 237*,

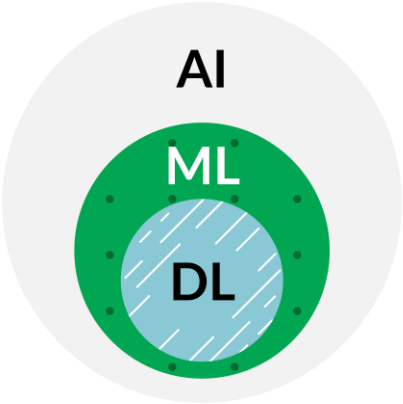
> 'The real danger of artificial intelligence is not that computers are smarter than us, but that we think [they] are'.

A common point of confusion to those new to the world of ML is how terms like artificial intelligence, machine learning and deep learning fit together.

Strictly speaking, ML refers to a subset of AI that enables a system to learn and operate from input data, rather than operating as a result of explicit programming. The other main branch of AI, often termed symbolic AI, relies on the explicit provision of logical statements, used to analyse and represent relationships between data, as opposed to learning from data as occurs in ML.

Deep learning (DL) is a further subset of ML concerned with a specific type of algorithm called neural networks (discussed in more detail below) which derive their name and structural inspiration from the human brain. Like their biological namesake, artificial neural networks are comprised of layers of interconnected neurons, each neuron capable of performing calculations, allowing neural networks to model highly complex relationships in the data.

Figure 1: Shows the relationship between deep learning, machine learning and artificial intelligence



Today, the terms ‘machine learning’ and ‘artificial intelligence’ are often used interchangeably, as ML has come to be the dominant AI technique. However, in order to understand a little more about ML, it is important to consider first the history and evolution of AI.

Coined in mid-1950s, the term AI was initially used to describe the process of solving logic problems and calculus on machines. Early programs were necessarily limited in scope by the speed and memory of processors and by the relative clumsiness of early operating systems and programming languages.

With a global increase in government funding for AI in the 1980s came increased development that paved the way for ML. New algorithms were developed that analysed data (what would now be called training data) by discarding seemingly unimportant data, in order to generate a set of general rules for the computer to follow. This approach would later switch to one determining the probability of certain outcomes based on the data, instead of simply using the data to determine rules.

The 1990s saw the development of computer programs that could analyse larger amounts of data, bringing with it a shift from a knowledge-driven approach to a data-driven approach producing predictive, or learned, outcomes.

Despite algorithms producing useful results, obtaining results was slow, with model training sometimes exceeding days or even weeks. Consequently, ML remained an academic endeavour and had little industrial impact. This didn’t change until graphical processing units (GPUs), specialised electronic circuits initially designed for the creation of images in gaming, started to be used to train models. This in turn vastly increased the amount of data that could be used for model training, which resulted in more accurate models.

New development tools and packages were designed, reducing computational barriers for academics in fields outside of computing. Github, a provider of internet hosting for software development and version control, allowed collaboration on projects from anywhere in the world, and huge databases such as ImageNet, a database with 14 million labelled images also became more prevalent, providing vast quantities of collated data for use in training ML systems.

In the early 2010s, frameworks, like Google’s Tensorflow and Facebook’s (as it was then called) PyTorch, that allowed users to conveniently build complex deep neural network models such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) (explained in more detail below) were released on an open-source basis.

Both the tools and the data required for developing useful ML models were now available to individuals outside of specialised ML groups in academia.

More recently, in the natural language processing field, we have seen large model training such as BERT from Google and its relatives such as GPT-3 from Open AI and ERNIE from Baidu. These have been made possible by significant investments by private companies in compute resources, using ever larger collections

of computers and processors. These larger frameworks proved that (provided there is enough data)—scaling up model size can lead to better task performance, producing often surprisingly human-like text.

In July 2021, Google-owned Deepmind introduced Perceiver IO, a more general version of BERT, capable of producing 'a wide variety of outputs from many different inputs, making it applicable to real-world domains like language, vision and multimodal understanding as well as challenging games like StartCraft II'.

For any neural network, the training phase of the model is the most resource-intensive task. Operations that would take years to perform sequentially can now be performed simultaneously using GPUs and other specialised processors, making the training of complex networks not only possible but fast and easy. The increased availability, and ability to store and process, data have been paramount to the increased success and applicability of ML. In a new economy driven by ML technology, the importance of data to fuel it is ever increasing.

The importance of data

We discussed above how increased processing speed has allowed models to be trained on vast amounts of data in a relatively short period of time. As a result, data is now often the limiting factor in model quality. The type and amount of data will determine the type of ML model used, while the quality of the data will determine the quality of the result. You only need to look at the discounts offered by supermarkets in exchange for tracking your purchasing history via club cards, to start to understand the value of good quality data.

Prior to model training, data analysis and exploration is typically carried out in order to better understand the collected data.

Data collection

An ML model learns using the data provided to it, so any biases in the dataset will be reflected in the trained model. Also, where a training dataset is too small, a model may learn only that dataset, and won't be able to generalise, using an inferred model. It will be good at predicting the data it has seen but will produce significant errors when asked to interpret new data. This is called overfitting and is discussed further below.

Data analysis/visualisation

Data analysis and visualisation are often used prior to training the model in order to understand the data, verify the quality of the data or decide how much of a given dataset to use. Visualisation techniques such as correlation matrices, line plots, scatter plots, histograms and box plots can provide insight into the most valuable variables in a dataset.

Data exploration and transformation (also commonly referred to as pre-processing)

Data exploration and transformation are used to ensure all the data is in the correct format to be provided as an input to the model. This may involve encoding written data so that it can be processed by the model. In addition, transformation may include filling in missing values, normalising the data, eliminating duplicates or removing or treating outliers where necessary.

The importance of data quality cannot be overstated. Poor-quality data will not only render a model useless, but may have dangerous consequences. For example, there have been reported instances of algorithms used for predicting the sentencing of prisoners, trained on historical court data, giving out higher sentences to people of colour than to white people with equivalent crimes. In such cases, the training data was found to contain a historical bias, which was then perpetuated by the algorithm.

It is important to remember that ML tools are not intelligent. They are capable of learning our mistakes but they cannot learn from them.

Training an ML model

ML models are designed and trained for the specific application for which they are used. Once developed and trained the computer then executes the algorithm. It is a common misconception to assume that you can simply give an ML algorithm a problem to solve and let the algorithm figure out how to solve it. In reality, ML

is just a different way of computing. The algorithm has been designed and trained for the specific problem for which it is used and the computer simply executes the algorithm that has been developed.

Training a model is the process of determining values for the parameters of the model in order to produce an accurate prediction from the input, or training, data.

Where the model returns a bad prediction there will be a penalty, which is described as a 'loss' in ML. The loss of an ML model is a number that indicates how badly the model's prediction performed on a single example. The penalty or loss should be minimised with a loss of zero indicating a perfect prediction. The goal of training is to find a set of model parameters that have a low loss on average across all examples.

Training an ML model, involves the following key stages:

- **defining the problem**—prior to training, the problem to be solved is identified and the potential inputs and outputs that would be required are considered
- **data collection**—data for training the model is then collected. As discussed above, the quantity and quality of data provided to the ML algorithm greatly affects the quality of the output
- **model selection**—both the desired outcome and the data that will be provided to the algorithm are considered in model selection. (Common ML algorithms are discussed in more detail below)
- **preparing the data**—characteristics and attributes of the data are selected and the data is pre-processed, for example to remove outliers (values that are significantly different from the main dataset), remove or fill in missing values, scale and/or offset the data so it fits the assumptions of the model to be trained
- **training**—the training dataset is provided to the model and the loss is minimised (as explained above) in order to improve the prediction rate and ascertain the model parameters that produce the best outcome
- **test**—a new dataset, consisting of data not used for training, is provided to the model to see how the model performs on new data it has not been trained on. If all goes well, the model should provide good predictions for the new data

Types of ML

The most prevalent type of ML is supervised learning, where the correct answer for the model output is used as a target for learning.

The other two main types of ML are unsupervised learning and reinforcement learning. Unsupervised learning looks for patterns in data, typically by finding a statistically efficient representation of the data. In reinforcement learning, a correct output of the model is not known or provided, but instead the model is provided with broadly defined desired outcomes. The model works by seeking to maximise rewards for meeting the desired outcomes.

Supervised learning

With supervised learning the model is provided with input-output pairs. It learns by mapping input training data to outputs based on the example input-output pairs that it has been given. The correct output is known and is used to train the model. A supervised learning algorithm analyses the training data and produces an inferred model that can be used to map new inputs to predicted outputs.

For example, a supervised ML model for predicting house prices would require a dataset of house price related information. The model's data input values might include, for example, distance from the city centre, number of rooms, proximity to schools etc, often called feature vectors. The model outputs would be house

price. The algorithm would be provided with input-output pairs (houses with different feature vectors as against their price). This would build a model which, upon input of a new feature vector for a new house, would output a predicted house price for the new house.

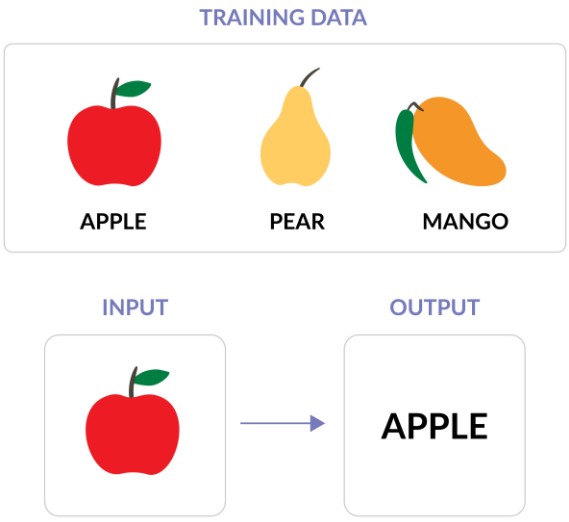
In supervised learning, the dataset is composed of examples; where each example has an input element (feature vector) that will be provided to a model, and an output or target element (house price) that the model is expected to predict.

Supervised learning can be further divided into two subcategories of ML:

- **classification**: seeks to assign a label to an unlabelled example. Classification learning algorithms take a collection of labelled examples as inputs (as shown in Figure 2) and produce a model that can take an unlabelled example as input and either directly outputs a label or outputs a number that can be used to deduce the label. In a classification problem, the label is a member of a finite set of classes, such as 'spam', 'not spam' in the case of binary classification, or three or more classes in the case of multi-class classification. Examples of common classification algorithms include: linear classifiers, support vector machines (SVM), decision trees, k-nearest neighbour, and random forest

- **regression**: seeks to predict a real-valued output (often called a target) given an input. The house price prediction mentioned above is an example of a regression task. A regression learning algorithm learns to output a continuous dependent variable given one or more continuous independent variables. It is commonly used to make projections, such as for sales revenue for a given business. Linear regression, logistical regression, and polynomial regression are popular regression algorithms. Neural networks are regression algorithms and are probably the most widely used today

Figure 2: An example of classification



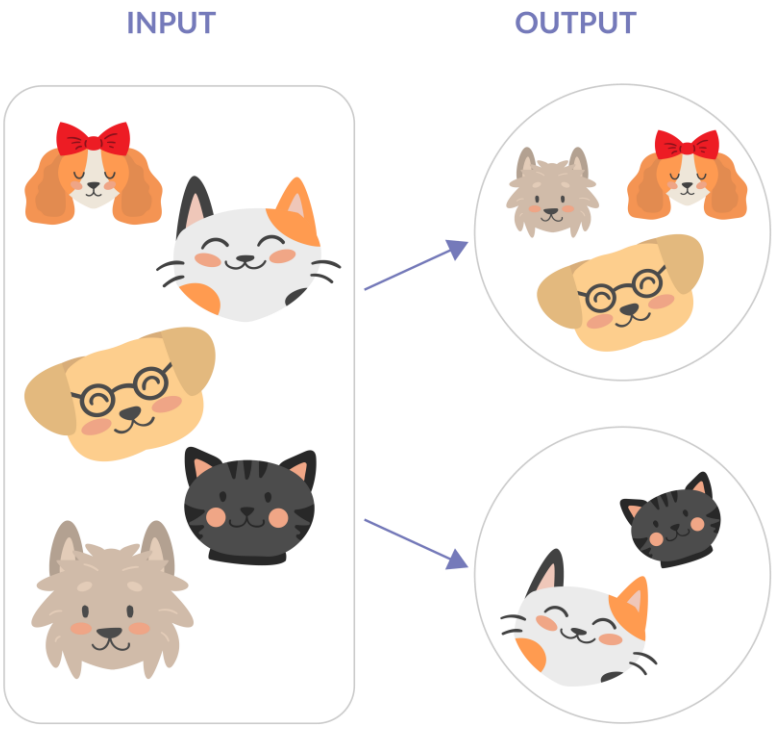
Unsupervised learning

Unsupervised learning uses a model to discover patterns or associations in data. The goal is to create a model that takes a feature vector and transforms it into an output that represents patterns or other information in the inputs in an interesting way. An example of this would be a model that is trained to represent pictures of handwritten digits learning that there are ten distinct patterns without being given any information as to what is in the pictures.

Unsupervised learning is particularly useful when subject matter experts are unsure of what the common properties within a dataset are. In contrast with supervised learning, no example outcomes are required when training unsupervised learning model. The model is left alone to discover patterns in the input data. In unsupervised learning, the training data is a collection of unlabelled examples from which patterns are discovered.

Clustering is a commonly used unsupervised technique. Clustering is the process of clumping data into clusters to see what groupings emerge (if any). Each cluster is characterised by its data points, and a cluster centroid in the middle of the data points. The cluster centroid is the mean (average) of all data points that the cluster contains. Clustering returns the properties of a cluster and the feature vectors that belong to each cluster.

Figure 3: An example of clustering images of animals into groups of similar types of animals



Common clustering algorithms are hierarchical (such as agglomerative hierarchical clustering, k-means, and Gaussian mixture models).

Semi-supervised learning

Semi-supervised learning is, as it sounds, a combination of supervised and unsupervised learning. The dataset contains both labelled and unlabelled examples. The goal is the same as the goal of supervised learning with the hope that using as many examples as possible, including unlabelled examples, will help the learning algorithm to find a better model or allow the user to deduce new information from the dataset.

Reinforcement learning

In reinforcement learning an ‘agent’ learns to operate in an environment using feedback. Unlike supervised or unsupervised learning, which typically both work with static datasets, reinforcement learning works with a dynamic dataset. This means that new data is continuously received by the model (the data provided to the model is continually changing) about each state.

States are representations of the current environment of the task. In reinforcement learning the agent learns by making a decision at each state and receiving a reward (or punishment) for that decision with respect to the final result, in order to learn the series of decisions that produce the optimal overall outcome.

For example, if you wanted to use reinforcement learning to teach a robot to walk to work, one state might be arriving at a particular road. A decision could then be whether or not to check for cars. The outcome of not checking could be an overall quicker journey to work (particularly if it is a road with very few cars), alternatively the outcome could be the robot getting hit by a car. Therefore, the algorithm would learn the optimum set of decisions based on the overall outcome of arriving to work.

Batch reinforcement learning (sometimes called offline reinforcement learning) is a variant of reinforcement learning where the agent learns from logged data, such as data from previous experiments. The same principles as above apply, except the result of each state is already in the static dataset.

Importantly, the power of reinforcement learning is that the model does not need to be told what the right decision is for each state the agent finds itself in. Instead, all that is needed is a definition of a successful outcome, such as arriving to work safely.

Considerations when selecting or assessing an ML algorithm

A key factor that is considered when selecting an ML algorithm is its performance on the dataset at hand. Models may perform poorly for a number of reasons. Perhaps the most common beingpoor-quality data.

Another common reason for poorly performing algorithms is overfitting. Overfitting is a term used to describe the situation where the model predicts the training data very well but performs badly on new data. Overfitting often results from models with a large number of parameters but a small number of training examples. The model learns the ‘noise’ in the data rather than the underlying structure.

Underfitting is used to describe a poorly performing algorithm where the model is unable to predict the data it is trained on. Underfitting often results when the model is too simple for the data or the features are not informative enough.

As well as model performance per se, other factors that are often considered and discussed when explaining algorithm selection are:

- memory and processing requirement (linked to cost)
- number of feature/examples
- type of data—categorical/numerical
- nonlinearity of data
- training speed
- prediction speed

Neural networks

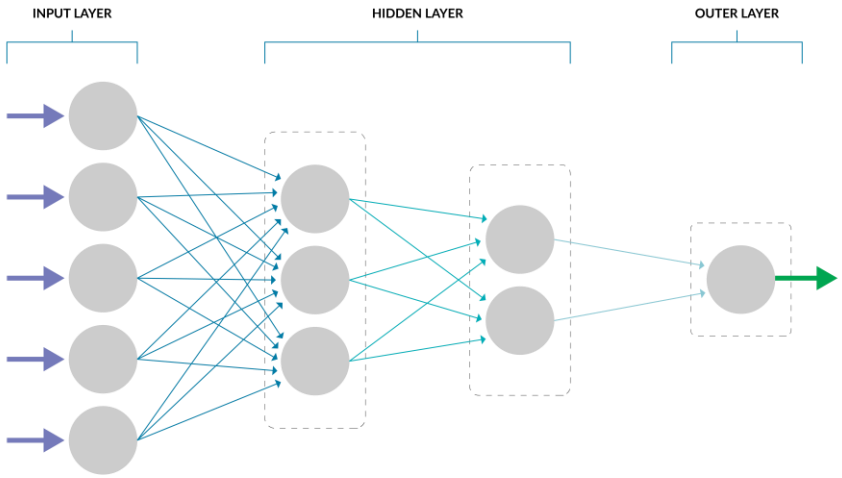
Neural networks, in particular, deep neural networks are by far the most prevalent ML technique. Neural networks derive their name and structural inspiration from the human brain. The term ‘neural network’, both in AI and in the brain, refers to a system of neurons. An artificial neural network does not depend on the capability of individual neurons. Each neuron performs simple mathematical functions. Rather, the effectiveness of an artificial neural network is attributed to the very large number of neurons and the complexity of the resultant network.

In its most simple form, a neural network is composed of three layers (with each layer being made up of neurons):

- an input layer
- a middle layer (referred to as a hidden layer), and
- an output layer

In reality, the simple form is rarely used, and more common are neural networks with multiple hidden layers. This type of neural network is called a deep neural network and is illustrated below.

Figure 4: Example structure of a deep neural network



In the illustration above, each neuron is represented by a circle. The inbound arrow represents an input of a neuron and indicates where the input came from. The outbound arrow indicates the output of a neuron. The output of each neuron within a dashed box, is the result of a mathematical operation.

Within each neuron in a dashed box, the following is happening:

- all the outputs from the neurons in the previous layer are combined to form inputs for the neurons in the next layer
- each neuron in the next layer sums these inputs, giving each input from the previous layer a connection weight that tells you how strongly the previous layer neuron is connected to the next layer neuron
- each of the next layer neurons applies the connected weight through a function called an activation function to obtain an output
- these outputs are combined as inputs to the next layer and so forth

The weights applied by the neurons are the parameters that define the network and are what is adjusted during training—these parameters are what is learned from the data. The information being learned about the data is contained in the connection weights of the network.

The architecture of the network does not change during learning, however, the output values of each neuron change with each input to the network. In other words, the memory of the network is in the connection weights and not in the neurons. This can seem counter-intuitive but it is precisely how we believe the brain works.

What is deep learning?

DL is a subset of ML using deep neural networks (neural networks with numerous hidden layers). The use of more than one hidden layer in a neural network provides a larger number of free parameters due to the increased number of connections between neurons and allows the network to learn levels of abstraction from one layer to the next. Again, this is akin to how the brain works—for example, your visual cortex has areas that react to lines and corners in images that connect to areas that react to shapes made up of lines and corners, and so forth. The result is a highly complex model capable of modelling complex data.

Common neural network architectures

There are many different types of neural network architectures, each suited to different learning tasks. The most common types are: feedforward neural networks (FFNNs); RNNs; and CNNs.

Feed-forward neural networks

The most frequently used illustrative example when discussing neural networks, is that of the FFNN. The DL neural network illustrated and discussed above is an FFNN. Feed-forward simply refers to the direction that the information moves through the layers. The simplest type of an FFNN is a fully connected FFNN, in which all neurons are interconnected from one layer to the next.

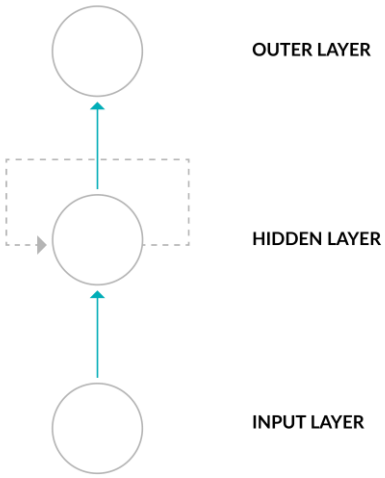
Convolutional neural networks

A CNN is a type of FFNN that reduces the number of connection weights by structuring how neurons are connected, organising the connections into groups called ‘receptive fields’—another term borrowed from the study of the brain. CNNs have applications in image classification and object recognition tasks since they reduce images into a form that is easier to process without losing features that are critical for getting a good prediction. The receptive field structure of CNNs is loosely modelled on how the layers of our visual cortex interconnect.

Recurrent neural networks

RNNs are used to label, classify or generate sequences. RNNs are often used in text processing because sentences and texts can be thought of simply as sequences of some sort (words/punctuation/characters etc).

An RNN is not a feed-forward network. It contains loops. Each unit of a current network has a state (or memory) such that each unit receives a vector of states from the previous layer and a vector of states from the same layer at the previous time in the sequence. This allows RNNs to produce predictive results in sequential data.



Transformer networks

Transformer networks are rapidly gaining popularity and have overtaken RNNs in the natural language processing arena. Like RNNs, transformer networks are designed to handle sequential data. Unlike RNNs, they do not process the data in order and avoid loops by processing a text sequence as a whole. In other words, transformers look at all of a sequence of text at the same time. That makes them much more efficient to implement than RNNs because the operations can be done in parallel processing hardware (rather than sequentially one after the other), which speeds up computation.

The magic ingredient in transformers is 'attention'. The network learns which part of the sequence to focus on, or, in other words, where to look for context for any given part of the sequence. This is akin to us paying attention to certain parts of the text in a sentence for context, rather than reading all the words in the sentence all at the same time.

Using deep learning

The big advantage of DL over other types of ML is its ability to handle large amounts of data. Unlike other types of ML where learning converges at a certain level of performance when you add more training data to the algorithm, a key advantage of DL networks is that they often continue to improve as the size of the dataset increases, provided that there are sufficient (huge amounts) of connection weights to adjust. The most powerful DL networks used today have hundreds of billions of connection weights, many orders of magnitude more than the number of parameters typically available to tweak in other types of ML.

DL has a multitude of applications in almost every field and is particularly popular in the health care industry, the finance sector and in image recognition.

However, there are recognised concerns connected with the use and deployment of DL technologies. By its nature, there is often no clear way to investigate, undo or discover the reasoning behind each prediction made in DL. This makes it difficult to use in fields where there are legal requirements to provide justifications or reasons for decisions that have been made—such as the granting or rejecting of loans. Further, the trend to ever larger models means that ever increasing amounts of computing power is needed to train these models, meaning that state of the art DL is a game dominated by private sector tech giants.

Some examples of other commonly used ML algorithms

Outside of DL, other approaches can yield powerful results with far less need for computing power. Some of the most frequently used supervised ML algorithms include:

- linear regression
- logistic regression
- decision tree learning
- k-nearest neighbours

Unsupervised learning algorithms mostly include clustering algorithms for practical applications. A general introduction to clustering is covered below. Reinforcement learning is another exciting field of research but further discussion of it is beyond the scope of this Practice Note.

Linear regression

Linear regression is a supervised learning algorithm where a model is trained to predict a value of a dependent variable 'y' for a given independent variable 'x', often a measurement of some kind. Since it is a supervised model, labelled examples are required to train the model, ie. pairs of x and y values are used for the training data.

Example problem: Learn to predict the price (y) of a house given information about the house (x) such as the number of rooms (x1), distance from a city (x2), square ft (x3) etc.

Logistic regression

Unlike linear regression, logistic regression is a classification learning algorithm. The goal is to learn a value of 'y' for a given 'x' where 'y' is the probability of one or more classes or outcomes.

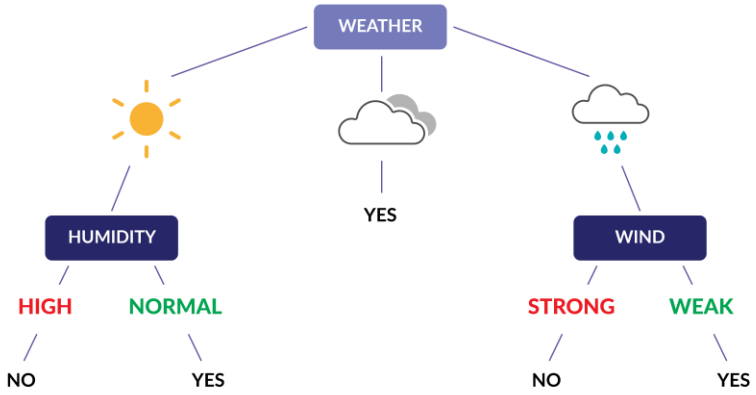
Example problem: Given x hours of study, predict how likely it is that the student passes an exam (y is the probability of passing).

Decision tree learning

A decision tree is a data structure that can be used to make decisions. Each node of the decision tree presents an attribute or feature and the branch from each node represents the outcome of that node. The algorithm identifies ways to split data at each attribute in order to predict the value of the target variable by learning simple decision rules inferred from the data features and evaluating how good each split is.

Example problem: Predicting whether or not your partner will go on a run given the weather forecast. The ML receives a feature vector (set of numbers) describing the weather conditions—wind strength, humidity, cloud coverage, rain etc. Passing those feature vectors through the decision tree will produce the decision output.

Figure 6: Example of a decision tree to predict whether or not your partner will go on a run



Random forests are a learning model which uses a combination of many decision trees. Typically, each individual decision tree in the random forest outputs a weak prediction and the prediction from all trees are combined to produce a strong prediction. For example, each tree can 'vote' for its prediction and the prediction with the most votes is the output; the prediction for the random forest.

K-nearest neighbours

K-nearest neighbours algorithms work on the principle that similar things exist in close proximity. New objects are assigned to the class most common among their k-nearest neighbours or are given a property value based on values of their k-nearest neighbours. K is a number that tells the algorithm how many neighbours to look at. Unlike the above algorithms, k-nearest neighbours keep all training data in model memory and use the data each time it is asked to output a class or property value for a new object. Often, what makes or breaks the algorithm is the quality of the feature vectors—the numbers used to characterise the objects. If the features are chosen so that objects similar in the property of interest group together, and are far away from objects that differ in that property, k-nearest neighbours can work well.

Example problem: Classify an image of an animal that looks similar to a cat or a dog into either the cat or dog class. Feature vectors that describe eye shape, body shape, sharpness of nails/claws etc would work well here but, for example, fur colour would not as it is not a feature which would effectively differentiate as between cats and dogs.

Clustering

Clustering is an unsupervised learning algorithm used to find structure in unlabelled data. It can be thought of as the separating of objects into groups whose members are similar in some way. A cluster is therefore a collection of similar objects.

Example problem: Finding groups of people with different characteristics in the population based on their online behaviour for marketing purposes.

Key challenges for AI and ML—transparency, explainability and bias

For some types of AI system there may be no easy way for a human to understand how the system reached a particular conclusion. It may not be possible for humans to review the decisions made by the system either before they are affected, or after (eg where the underlying data, code or algorithms are withheld on the grounds of trade secret or proprietary information). Often, AI solutions are referred to as 'black boxes', as the algorithms and datasets on which they are programmed are not open or available for review or audit. There is much research and development going into dealing with this problem. The field of research is often called 'Explainable AI'.

As well as being difficult to discover the logic behind a particular action or decision made by the system, there is also a balancing of commercial interest at play. Commercial organisations invest a lot into development of AI systems and are keen to protect their investment and their exposure to liability. If making a system explainable and decisions transparent means exposing how a system is designed, this may be contrary to an organisation's commercial interest. That said, explainability and transparency are some of the key ingredients that will be needed for wider adoption and acceptance of AI technology, so it will be interesting to see how this tension plays out.

Because AI systems learn to make their decisions based on the training data that they have been given, any bias which may be included in that data, will also be learned by the algorithm and reflected in the subsequent model. Even if obvious variables are removed from the input, such as gender, race or sexual orientation, other, more subtle, social or historical inequalities may remain. For example, Amazon stopped using a hiring algorithm when it became clear that it was favouring applicants based on words like 'executed' or 'captured', being terms that tended to be more commonly found in men's CVs.

Bias can also find its way into an AI system through flawed data sampling (sample bias or selection bias), where certain groups are over or under represented in the training data. For example, facial recognition algorithms trained mainly on Caucasian faces will not work well for ethnic minority faces. If such algorithms where used to control access to, say banking or other services, this could perpetuate social exclusion.

Labelling bias occurs where there are inconsistencies in the labelling process. This type of bias commonly arises when different annotators give different labels to the same type of object (for example, sofa and settee). It may also occur where the annotator applies a subjective (as opposed to an objective) label to the data, reflecting subjective preferences of the annotator.

Negative set bias occurs where there are not enough samples given to define what a phenomenon 'is not' (negative instances), only examples of what a phenomenon 'is' (positive instances). As a result, the algorithm may struggle to identify negative instances as easily as positive.

Human decision making is also likely to be biased but AI provides an opportunity to improve on traditional human decision making. Where variables do not accurately predict outcomes in the available data, ML can learn to disregard them, in contrast to humans who may be oblivious to the factors that led them to make a decision. It is also possible to probe an AI system to discover bias. Even where DL technologies are used, it may be possible to obtain more information about the decision made than it would be to uncover our own unconscious biases.

In November 2020, the Centre for Data Ethics and Innovation (CDEI) published its review into biases in algorithmic decision-making, recommending that the UK mandate transparency obligations on public sector organisations with respect to decisions made about individuals.

The report recognised the difficulties inherent in navigating the analytical techniques necessary to expose and identify bias. It recommended that regulators and industry bodies work together to identify best practice in their industry and to establish appropriate regulatory standards.

The CDEI review into bias in algorithmic decision-making stated as a core theme of the report, the 'ethical obligation to act wherever there is a risk that bias is causing harm and instead make fairer, better choices'.

The report recognised the difficulties inherent in navigating the analytical techniques necessary to expose and identify bias. It recommended that regulators and industry bodies work together to identify best practice in their industry and to establish appropriate regulatory standards.

In November 2021, the UK government published an algorithmic transparency standard for government departments and public sector bodies, promoting transparency in the way in which algorithmic tools are used to support decisions, particularly decisions with a legal or economic impact. The standard was accompanied by a new template and guidance on how to use algorithmic tools to support decisions.

Privacy and data protection

Under the General Data Protection Regulation (GDPR), there are specific requirements for the provision of information about and the explanation of AI-assisted decisions. Other key data protection principles will also, by necessity, require explainability for compliance. The Information Commissioner's Office's (ICO) Explain Guidance sets out practical guidance for compliance with the UK GDPR's explainability requirements. The ICO has also produced Guidance on AI and Data which looks at explainability of AI from an audit and compliance perspective. As a general point, it notes that 'black box' models (typically based on DL where the logic of the system may be difficult to explain) should only be used where:

- the potential impact and risks have been thoroughly considered in advance and the use case and organisational capacities/resources support the responsible design and implementation of such systems, and
- the system includes supplemental interpretation tools that provide a domain-appropriate level of explainability

Protecting AI technology

There is considerable investment and value at each stage of the AI development and deployment, and protecting that investment and the ownership of any rights in the AI is a key factor. AI challenges protection through traditional intellectual property mechanisms in numerous ways, raising niche issues depending on the type and functionality of the AI in question. Detailed consideration of those challenges is beyond the scope of this Practice Note.

RELX (UK) Limited, trading as LexisNexis®. Registered office 1-3 Strand London WC2N 5JR. Registered in England number 2746621. VAT Registered No. GB 730 8595 20. LexisNexis and the Knowledge Burst logo are registered trademarks of RELX Inc. © 2022 LexisNexis. The information in this document is current as of September 2022 and is subject to change without notice.