Table of Contents

Summary

What is Machine Learning, exactly? Is it the same as Artificial Intelligence? These terms have exploded in common and industrial usage in the past decade, and it is increasingly important to be able to understand and talk precisely about these topics. ML promises to offer firms a new paradigm of automation, yet is often less transparent than other methods. This means that significant care must be taken to avoid undetected mistakes leading to misplaced trust in automated systems that are not performing as expected. Similar problems can arise from data alone if existing human biases are captured by these algorithms. For these reasons, this document will guide engineers to:

- Demystify the jargon surrounding AI and Machine Learning
- Navigate the vast amount of ML theory, techniques and resources available
- Solve problems, avoiding common mistakes and wasted efforts

It can be used as a reference before, during and after a machine learning project is developed.

Introduction

“Can machines think?” Was the question posed by Alan Turing in 1950 to open the field of Artificial Intelligence. He challenged the world, and himself, to replicate natural (human, animal) intelligence artificially, creating systems that can think and act rationally. Creating a system that could achieve this to the level of a human would be an Artificial General Intelligence (AGI), or a “strong” AI by some definitions. The answer to his question remains unknown to this day, however it is certainly possible for machines to learn. In the sense hat algorithms can adjust themselves to improve performance, “Machine Learning” (a.k.a “weak AI”) is not only possible, but in widespread usage today.

This forms the core distinction between strong AI and ML. Machine learning algorithms cannot “think” like humans can, but they can use data to “learn” to improve themselves. In this document we will use “ML” to refer to the class of algorithm described by weak AI and Machine Learning, and “AI” to refer to Strong AI and AGI.

Where AGI refers in some way to an artificial equivalent to the human brain, it is perhaps less intimidating to think of ML as any computer program which learns through self-adjustment. The gist of ML is that we write a generic algorithm with a list of settings that determine what the algorithm does. Initially, the settings are decided randomly, so the algorithm won’t do anything useful. But then, we can use data to optimise (tweak) those settings to make the algorithm do what we actually want it to do. Simple!

A great number of techniques come under the umbrella of ML. These will be broken down in detail later in this document, however an important category to consider is called “deep learning” (DL). Yet another term often (mistakenly) used interchangeably with AI and ML, this refers to a specific subset of ML algorithm, called deep neural networks. Whereas non-DL methods usually expect their input to be made of individually useful elements (think eye colour, jaw length, glasses(Y/N) etc) called “features”, deep neural networks are used to learn to extract useful features from high dimensional data (think the individual pixel values from a photo of a face).

Language frames the way we think, thus it is critical to have a strong understanding of these concepts and associated terminology. Having established here what we mean when we talk about AI, ML and DL, the first aim of this document is to clarify all the remaining technical jargon an engineer might encounter when considering using a machine learning approach to a problem.

Then, armed with this terminology, we can dive deeper into the details of how to approach building a robust and trustworthy machine learning system.

Overview: Key Concepts and Terminology

In this section, we introduce and overview some of the key concepts and terminology that are important in understanding and discussing AI and machine learning technology. This section is not intended to serve as an exhaustive discussion of the topics in question, but as an introduction to the key points and vocabulary that will frame the rest of the discussion on this topic. Key terms and concepts will be highlighted for the reader, with links to a more detailed description of the term in the appendix.

To briefly review what we discussed in the introduction: the type of intelligence and learning in algorithms that we are interested in in this manual are algorithms that learn mathematical models of the world from some data, in order to make predictions about other unseen or future data. One important idea that we need to consider first is structured data and unstructured data.

Breakaway: Structured vs Unstructured Data AI and Machine Learning models are no different from any other computer program in that they require their input data to follow a consistent format. Unfortunately, data collected in the real world rarely follows the type of structure and organization that is needed for ingestion by an AI or Machine learning algorithm, and in many cases the work done to transform data into an appropriate structured format is some of the most important work done in any AI and Machine Learning pipeline. We make this distinction between data that has been put into a useful structured format as structured data, and data that exists in a raw, unprocessed format as unstructured data.

When dealing with data in the real world, we will often split it up into categories or types. One such distinction often made that is especially important in the context of AI and Machine Learning is the split of data into continuous data and discrete data. Continuous data can take on any number of infinite values across a given range, for example, a measure of rainfall per hour. Discrete data on the other hand is any type of data that falls into a fixed number of categories. These categories can be both ordinal data in which there is a natural ordering between the categories (shoe size, for example), and nominal data, where the categories are distinct (eye color, for example). While this distinction is important for many parts of AI and Machine Learning, the distinction between whether an AI and Machine Learning algorithm is trying to predict continuous and discrete data is so important that it has its own nomenclature of regression and classification algorithms respectively.

While these distinctions are important, we suggest that readers focus on the practical and functional aspects of this distinction rather than its definitional aspects unless they are doing hard math. Consider, for example, the case of the price paid at a market for an apple. While it may be definitionally appropriate to consider the price as a discrete number (unless our merchant is willing to accept fractional coins), it may be practically more suitable to consider it as continuous. Avoiding such problems is why statisticians use measure theory, which we caution the reader is even less fun than it sounds.

Breakaway: Regression vs Classification Algorithms The distinction between regression (continuous output data) and classification (discrete output data) is particularly important in AI and Machine Learning algorithms, because the type of data that the algorithm outputs has a significant effect on how it must function. Notably, some algorithms (e.g. Support Vector Machines) are only designed to function in one of these modalities, and require significant adaptations to perform (likely very poorly) in the other.

While we have been discussing some of the concepts and terminology around data to this point, we have used the terms “learn”, “learning” and “learning from data” to describe what our algorithms do without really making it explicit what we actually mean by this. One of the reasons that we’ve avoided doing this is that “learning” in the context we’re discussing it is conveniently, without further qualifiers, a term that covers several different ideas. These differences stem from the way that we use data in order to “learn”. The most prominent of two of these ideas are supervised learning and unsupervised learning, which are concerned whether we learn from data that list the correct output the algorithms should produce for some given input data (labeled data), or simply the input data themselves (unlabeled data).

Breakaway: Supervised vs Unsupervised vs Reinforcement vs Other Learning We use the nomenclature of Supervised vs Unsupervised (vs others) to describe the way in which our algorithms are learning. In Supervised learning, we learn from matched input data/output data pairs, data for which we already have the correct output the algorithms should predict for a set of given inputs (“learning by example”). We call this data labeled data, because our set of input data is labeled with the corresponding correct solutions.For example, we might be interested in predicting the future prices of the stock market from economic indications, by looking at how these economic indicators have predicted its historical past prices. In Unsupervised learning, we only have access to the input data without any corresponding output solution attached. We call this data unlabeled data, and our unsupervised learning algorithms and are generally interested in predicting some quality of this data (“pattern learning”). For example, we might be detecting unusual anomalies of electrical usage in the grid.

While it is generally preferable to use supervised learning when we can because learning by example is easier, there are many situations in which unsupervised approaches are more appropriate. Even putting aside the fact that unlabeled data is easier to collect (since we don’t need to label it), for many problems supervised approaches are simply not practical. In our electrical grid example above, it would be infeasible to train a supervised model to do similar anomaly detection. By definition, anomalies are rare and unusual data points that fall outside of the usual observations in the data. Creating a labeled dataset of them would be both impractical, and any supervised algorithm that used it would be prescriptive - it would only catch anomalies similar to anomalies we’ve trained on, where an unsupervised approach instead catches ones that are dissimilar to everything we’ve seen so far.

There are also several other learning approaches that fit within the supervised/unsupervised dichotomy discussed so far. A common one is Reinforcement Learning. In Reinforcement Learning, the algorithm is not fed a set of data, but selects which piece of data it wants to learn from in future from the pieces of data it has had up until now. Another common paradigm is semi-supervised learning, in which an algorithm learns from some set data that is labeled, and some (usually larger) set of data that is unlabelled.

AI/ML Overview and Trade-offs

Aims of this document:

- Key considerations to be aware of in employing AI/ML in an embedded system
- Guidance towards application appropriate solutions given relevant trade-offs and system-level requirements

- Doesn’t appear to be anything like this online
- Who are the end users and how will they use it?
- Tim likes “cheat sheets”
- How affected by the type of users (Verification engineer, designer, …)
- Only industry or also academic (note specific to TechNeS, and aimed a professional engineers)
- Can we reuse academic courses
- How much more detail needed
- What prior experience assumed (general and AI specific)?
- Make it narrower, focus on applications, examples specific users
- Should I even use AI?
- Need decision steps, choice of algorithm is towards the end, may need to try several
- different collateral for stages as user gains expertise
- Start with good and bad examples of AI in use
- how to balance information overload: break into steps, separate guides for different problems

Candidate “Documentation” Technologies

Requirements

- Must be good for collaborative work
- Must generate at least PDF, HTML, ideally help files
- Must be able to add mixed media (images/video)
- Needs version control
- Must be freely available to the group

Candidates

- Restructured Text
- DocBook/DITA

Non-candidates

- Overleaf/LaTex - can’t generate structured HTML
- MS Word/OpenOffice/LibreOffice - can’t generate structured HTML
- Canva - proprietary

Top Level Decision steps

Step 1:

Train user in vocabulary to be able to address the problem accurately. Enable conversation between engineering and marketing to refine product (may be highly disruptively).

Step 2:

Can you relate your problem to an existing problem AI has already solved (data architecture and algorithm). Or do you need a new AI approach. Is it possible with AI? Or do you even need AI? We need to create the inverse table: what AI can do. Allow user to refine from general to specific AI choice

Step 3:

Identify candidate solutions (data architecture and algorithm), cross-check features against table in this document. Do this hierarchically. Identify any wider impact on the system by choice of AI.

Step 4:

Identify AI engineering skills needed (training/buy-in)

End step:

Engineer can implement their project

Overview: How Algorithms Learn

How AI Algorithms learn	Description
Supervised	**The model is trained on 'labelled' data, where known correct outputs are labelled alongside the input data to enable future classification or predictions, sepcialized variants** • Semi-supervised: Not all the data is labelled or reliable, so need to mix with unsupervised techniques to pre-process the data • Self-Supervised: Applying a pre-trained (pretext) supervised learning model to an unsupervised learning problem to address a specific problem
Unsupervised	The model is presented with unlabeled data and so has to exploit inherent qualities in the input data, such as clustering or density / distribution to 'learn' how to process it.
Reinforcement	Technically a class of supervised learning, but usually categorized separately. The model operates without training but in an environment where it's decision outputs (based on given input data) will elicit reward signals (positive and negative feedback). It must thereby optimise it's internal model of likely success from a given current state.

Neural Network Algorithm Families

AI Algorithm Families	Description
Feed Forward Neural Networks	A collection of layered, interconnected nodes which sum weighted inputs and applies an activation (mathematical) function to derive output. Input data travels in one direction only and exits through output nodes. Before use FFN's are trained using a training dataset and thus use a supervised learning approach.
Recurrent Neural Networks	By feeding intermediate layer outputs back to the inputs, better prediction of outcomes is achieved. In this way, some information the network possessed in the previous time- step is remembered by a memory function.
Unsupervised Neural Networks	The network does not receive a prior training dataset and, instead, is presented with unlabeled data. The network therefore has to exploit inherent qualities in the input data to 'learn' how to process it. Typically 'training' occurs 'on the job' e.g. using competitive rather than error-correction learning.

Graph Neural Networks	
Spiking Neural Networks	

Non-Neural Network Algorithm Families

AI Algorithm Families	Description
Linear and related	• Linear Regression • Spline Interpolation • Support Vector Machine (SVM)
Tree Based Methods	• Decision trees • Random Forests • Boosted/Bagged trees • Gradient Boosted trees
Nearest Neighbor	• K-nearest neighbor • k-means

Statistical	• Naieve Bayes
	• T-test/F-test
	• **Markov Chain Monte Carlo**
	  • Simulated Annealing
	  • Dynamic Causal Modelling
	• Full Bayesian Methods

Symbolic	• Inductive Logic Programming
Bio-Inspired	• Genetic Algorithms/Genetic Programming • Any Colony Optimization • Particle Swarm Optimization

Mathematical Models

Algorithm	Description	Type	Learning	Benefits	Application

Linear Regression	Line-fit algorithm, according to: y=ax+b.	Regression	Supervised	Efficient for linear relationships e.g. trends or forecasts	Drug-dosage relationships
Logistic Regression	Discrete outcomes from linear data.	Classification	Supervised	Efficient for linear relationships when the data set is linearly separable	Medical data, Credit scoring, Language processing
Decision Tree	Data set is split recursively into a tree comprising decision nodes and outcome leaves.	Classification (or Regression)	Supervised	Fast and efficient to run. Easy to understand and interpret. Can handle any type of data	Data mining, Planning, Fault diagnosis
Support Vector Machine	SVM algorithms classify data in n (no of features)-dimensional space.	Classification (or Regression)	Supervised	Handles high dimensional data, separating things into (typically two) groups with more separation than other algorithms by projecting the data into a more easily separable space.	Face detection, Handwriting recognition, Image classification
Naieve Bayes	Simple classification algorithm using conditional probability of an event based on prior events.	Classification	Supervised	Requires less training data than other algorithms and handles both continuous and discrete data. Highly scalable with the number of predictors and data points. Fast runtime enables real-time predictions.	Face recognition, Weather prediction, Medical diagnosis, News classification, Spam email detection based on word frequency vs real email.
K-nearest Neighbor	Classification based on prior classification of the majority of (K) nearest neighbors with a distance function. (Becomes computationally expensive as dataset and dimensionality scale).	Classification (or Regression)	Supervised	Simple to implement and effective for data with low dimensionality. There is no training period (data is stored only for use later) which makes KNN faster than other (trained) algorithms. Furthermore, new training data can be added seamlessly.	Text mining, Finance, Medical, Facial recognition, Recommendation systems (e.g. music based on age, genre, country)

K-Means	Classifies data into K clusters through recursive clustering of data with similar features.	Classification (or Regression)	Unsupervised	Simple to implement and results are easy to interpret. Handles large datasets well and guarantees convergence. Easily adapts to changes in data.	Image segmentation, Image compression, Biological data, Fraud detection, Transport data analysis
Random Forest	Average across multiple decision trees trained on various subsets of the data.	Classification (or Regression)	Supervised	Reliable predictions that can be understood easily. Handles large datasets efficiently. More accurate than a single decision tree.	Finance risk, Medical trends, Stock trading, E-commerce

Feed Forward Neural Networks

ANN	Description	Application
Feed Forward Neural Network	Collection of interconnected nodes, arranged in layers. The basic unit is the Perceptron (or Threshold Logic Unit) - a single node which sums weighted inputs and applies an activation function to derive the output. In larger networks with multiple nodes, input data travels in one direction, passing through a number of input nodes and exiting through output nodes. In a multilayer network (three or more successive layers), each node is connected to all nodes in the next layer. The output is a function (activation function) of the sum of all inputs multiplied by their respective weights. During training (Supervised Learning), the weights are calculated through backpropagation.	Data Compression, Pattern Recognition, Machine diagnostics, Image / Speech / Handwriting Recognition
Convolutional Neural Network	A form of FFN Inspired by the animal visual cortex. The CNN is a 3D arrangement of neurons, employing convolutional processing rather than multiplication in its hidden layers. Each neuron in the first (convolutional) layer only processes information from a small part of the input field. The network understands images in parts and computes these operations multiple times to process the full image. Nodes are connected only locally to nearby neighbors unlike an FFN.	Video recognition, semantic parsing and paraphrase detection.
Radial Basis Function Neural Network	A three-layer FFN where the hidden layer uses a non-linear RBF activation function. Classification is performed by measuring the input’s similarity to previously trained data points.	System modelling & control, time series prediction, image classification

Recurrent Neural Networks

ANN	Description	Application

Recurrent Neural Network	Layer outputs fed back to the inputs help in predicting outcomes. The first layer is typically a feed forward neural network followed by a recurrent layer where some information it had in the previous time-step is remembered by a memory function.	Handwriting / Speech recognition, Language modelling & translation, Text summarization, Image tagging
Long short-term memory (LSTM)	A more sophisticated RNN which uses memory gates to regulate information flow through the network so as to improve training by avoiding loss of small gradient data during backpropagation.	Unsegmented connected handwriting recognition, speech recognition, robot control, video games
Sequence to Sequence (Seq2Seq)	Two Recurrent Neural Networks working simultaneously. One RNN is configured as an encoder, processing the input data and the second as a decoder which derives the output based on the encoder's final internal state.	Machine translation, Speech recognition, Text summarization, Conversational models / Chat-bots, Video captioning
Attention network	Attention networks mimic cognitive attention by enhancing some parts of the input data while diminishing other parts; i.e. to focus attention on a small but important part of the data. Learning which part of the data is more important than others depends on the context and is trained by the gradient descent algorithm.	Reasoning, Complex language processing. Multi-sensory data processing (sound, images, video, and text)

Recurrent Neural Networks

ANN	Description	Application
Self-Organizing Map / Kohonen Net	Unsupervised technique producing a 1 or 2-D representation of a higher dimensional dataset such that similar observations are clustered to aid onward analysis. The network is trained using competitive rather than error-correction learning so that nodes 'move' within the dataspace to generate a map of the reference data. During iterative training, node weights change in order to 'cluster' the neurons together to reduce the distance between neuron and input. The map can then classify observations for the input space by finding the node with the closest weight vector to the input space vector.	Visualizing data in large datasets, Project prioritization, Seismic or Failure mode analysis, Artwork creation
Generative adversarial network (GAN)	Two neural networks compete in a zero-sum game. The generator network learns to generate new data with the same statistics as the training set in an unsupervised way, as it is indirectly trained by the discriminator (the second neural network) which can tell how realistic a given input is while it is itself being updated dynamically. GANs are also useful for semi-supervised, fully supervised and reinforcement learning.	Image-to-Image / Text-to-Image Translation, Semantic image manipulation and creation, Photo editing / blending, Face aging, Video prediction, Super-resolution

Autoencoder	An autoencoder is a Feed Forward Neural Network which learns an efficient representation (encoding) for a set of unlabeled data , through unsupervised learning. The autoencoder consists of two main parts: an encoder that maps the input into the code, and a decoder that maps the code to a reconstruction of the input. The encoding is refined by attempting to regenerate the input from the encoding whilst minimizing the difference between input and output. The network therefore generates new data rather than predicting target values and is thus unsupervised.	Dimensionality reduction, Information retrieval, Anomaly detection Feature extraction, Image denoising and compression, Image search, Image generation