ML Models: Understanding the Fundamentals

Introduction

Machine learning (ML) is a process by which computer programs learn to recognize patterns and behaviors in data. The goal of machine learning is to build computer models that can make predictions on new data or novel information. These models can then be used to automate repetitive tasks and make basic (and increasingly complex) decisions without human intervention.

To *teach* a computer program to build a model, we first need to design an algorithm, which is a finite sequence of well-defined instructions that solves a specific problem, such as calculating the probability of an event occurring or classifying an object. We then feed that algorithm a steady diet of data to ingest and process, so it can develop a catalog of examples (a model) to compare to any future inputs it receives. Generally, the more well-fed the model the better its performance will be, though good data sources are often hard to find.

In this post, we'll explore the fundamentals of machine learning, the various forms of models of the physical world that can be developed through this process, as well as the expanding list of human tasks that can be tackled using these trained machines. Finally, we'll show you how some newer machine learning models are designed to generate new, synthetic data that can be used for training other ML models.

What are ML Models?

While the concept of training machines is a bit abstract, ML model applications are everywhere in our daily lives. From your smartphone making text suggestions when messaging a friend, to Uber adjusting arrival times to destinations based on dynamic traffic patterns, and Netflix recommending your next show to watch – each of these tasks is executed using ML models that have created mathematical representations of their real-world processes.

ML models are a great tool to help predict future events. There are an increasing number of real-world tasks, decisions, and objects that can be represented mathematically as repeatable, step-by-step processes or pixel-by-pixel images, and then converted into code that a machine can act on. A model trained on historic data can discover past patterns that then inform inferences or predictions about what may happen next.

The Basic ML Model Architectures

Each machine learning algorithm settles into one of the following basic model categories, based on how it's designed and what type of data it's trained on:

- Supervised Learning
- Unsupervised Learning
- Self-Supervised Learning
- Reinforcement Learning

Let's briefly discuss each.

Supervised Learning

Supervised learning is a type of machine learning algorithm that learns from data with *known outcomes*. In supervised learning, your model is trained on *labeled data*, which means that each input in the dataset has been assigned a specific label or outcome as its output. This gives the algorithm a benchmark to aim for in its predictions and improve its accuracy as it learns. Over time, the goal or function of the algorithm is to adjust its error rate to map more closely to a generalizable desired result. There are two main types of supervised learning models: regression models and classification models.

Unsupervised Learning

Unsupervised machine learning algorithms are a great tool for learning hidden patterns in data. Here, you are not providing predefined classes for the model to sort data by, so the model will group the *unlabeled data inputs* into various clusters of derived values or features that it deems measurable, informative, and non-redundant. The model will then classify these values on its own. This unguided, free-association process enables the discovery of novel patterns that a supervised learning model would not otherwise be looking for or find.

Unsupervised learning models are used to perform three broad tasks:

- **Clustering** — finds the natural groupings for all data.
- **Association** — finds the dependencies or interesting relationships between various data.
- **Dimensionality reduction** — finds the intrinsic components that represent certain data.

Self-Supervised Learning

Self-supervised learning is a bit of a hybrid model. It leverages signals from the structure of the unlabelled data it is ingesting to create a supervised task where it predicts the unobserved properties of the inputs and discerns which data points are similar and which are different. With enough information, the model can develop a form of commonsense reasoning or generalized knowledge where it understands the meaning of certain information in specific contexts. The most popular types of self-supervised learning models are transformers.

Reinforcement Learning

In reinforcement learning, the algorithm can interact with its environment. Because it has a sense of agency in the world, these models are referred to as *agents*. An agent's feedback system is governed by a policy that rewards good actions and punishes bad actions and is optimized to guide this cyclical learning process towards an ideal desired outcome. These policies and ideal states can be supervised or unsupervised to seek specific rewards. This type of learning is most similar to theories of human learning, as we also learn things through experiences and feedback from our environment.

There are several popular algorithms that come under reinforcement learning. They all work in similar ways – by processing feedback signals after each current state or action taken by the model in its environment and then optimizing its subsequent actions based on that real-time, incoming information and its policy goals.

Below are the main RL algorithms and their policies:

- **Quality (Q)-Learning** — finds the best action to take given the current state.
- **State-Action-Reward-State-Action (SARSA)** — finds the best action to take to learn the Q-value using the action performed under the current policy.
- **Temporal Difference (TD) Learning** — finds the best action to take using changes or differences in predictions over successive time steps.
- **Deep Quality Network (DQN)** — finds the best action to take by determining values for multiple different possible actions, depending on the state of the system.

Next, let's delve into the common model tasks in more detail.

Types of ML Models

We've already briefly discussed two of the main categories of machine learning model tasks: classification and regression. Now, we'll take a closer look at how these models can be designed for more specific tasks.

Classification

Classification models are used to predict outcomes in a categorical form. They group input data into different categories and assign labels to each category. A common example is a model that identifies and labels an email as either spam or not, or a customer as likely to purchase a specific product or not. A more modern example, especially for developers and enterprises who are building products and services with large amounts of data is to detect and label any sensitive data that may be included in a dataset, so they can ensure user privacy is protected.

There are two types of classifications in machine learning:

- **Binary Classification** — where there are only two possible prediction outcomes or classes. For example, the answer is either yea or nay.
- **Multi-class Classification** — where there are more than three possible prediction outcomes or classes. For example, the answer is either behind door A, B, or C.

These two classification model types can be further broken down by approach. There are piles of classification methods, but here are three of the most common and how they work:

- **Logistic Regression** — finds a number or a probability (called a “logit”) that a given input is associated with a certain category. If the probability of the input variable is over .5%, for example, the model will infer it’s a specific class. The name is misleading but it is basically a classifier version of a linear regression, which we will discuss below.
- **Support Vector Machine (SVM)** — finds the best decision boundaries, or range of correct predictions, in an N-dimensional space. These boundaries can segregate inputs into multiple classes based on input attributes, like their size, shape, or value. This is a more accurate version of logistic regression. With advanced SVMs, you can also use kernels, which allow you to adjust the dimensions of the data it is trained on. This shift changes the perspective from which the model interprets the data. It’s like giving it rose-colored glasses to wear.
- **Naive Bayes** — finds the best probabilistic classification value for an input, based on the assumption that the value of a specific data elements or attributes is independent of any others present in the data. The model is ‘naive’ in the sense that it ignores conditional dependencies between these data variables. Calculating conditional probabilities for dependent and independent variables is a core component of machine learning.

Regression

Regression models use supervised methods to map the relationships between data variables by fitting any inputs to a defined geometric line (either straight or curved) or a plane (in the case of two or more independent variables). This allows you to estimate how dependent and independent variables change and interact over time. An example could be measuring the strength of the relationship between two variables like rainfall and temperature. Or you could measure a dependent variable at a certain value for a certain independent variable, like the average temperature at a certain level of rainfall.

Here are the main types of regression model algorithms:

- **Linear Regression** — finds the best-fit line that most accurately reflects the relationships between data points, then uses this line to output continuous predicted numbers for future values. It learns by using its *cost or error function* to adjust the *weights* it has assigned to each input variable until it has minimized its error rate. This statistical process is known as gradient descent, where the model starts on a steep slope on a graph (of high error) then improves, moving down the slope until it hits the "bottom of the bowl" (of high accuracy).
- **Decision Tree** — finds the most accurate prediction through the use of a tree-like structure of if/then decision steps, with the possible consequences and outcomes of each step leading to some final output. Each step or 'node' represents a test on a given input attribute, with each branch representing the outcome of that test. The more 'test' nodes a decision tree has, the more accurate the final output will be.
- **Random Forest** — finds the most accurate prediction by aggregating and averaging the predictions of a large number of decision trees. This method of combining different model outputs is known as 'ensemble learning.'
- **Neural Network** — finds the best predictions for numerical or categorical values by processing information through a multilayered structure that includes one input layer, one or more hidden layers, and an output layer. As each node or *neuron* is interconnected, the model transfers inputs from neurons in one layer to those in the next layer, until it finally reaches the last layer of the network where it generates an output.

Clustering

Clustering is a great way to group inputted data together based on similarities and differences. Clusters correspond to connected areas in an N-dimensional space where there is a high density of data points. The similarity of any two data points is determined by the distance between them, and there are many methods to measure the distance. These clusters should be 'stable' in that they should be possible to characterize using a small number of variables.

These models can be used for a variety of tasks like image segmentation, statistical data analysis, or market segmentation. Here are some of the various types of clustering models:

- **K-means** — clusters inputs so that data points in the same cluster are closer together (i.e., similar) and data points in different clusters are farther apart.
- **Hierarchical** — clusters similar data points that have a predominant ordering or sequence from top to bottom.

- **Mean-shift** — clusters data points iteratively by shifting each point towards the densest area of data points i.e. the *cluster centroid*. Unlike K-means, this model does not need the number of clusters specified in advance because the number of clusters will be determined by the algorithm as it trains.
- **Density-based** — clusters data points by assuming a cluster is a region of high density in dimensional space that is separated by regions of low data point density. This method has the advantage of discovering arbitrarily shaped or hidden clusters.

Dimensionality Reduction

Dimensionality reduction is an unsupervised learning technique used to reduce the number of features or variables present in a dataset. It is the transformation of data from high-dimensional to low-dimensional space, through the deletion and combination of certain features so that the low-dimensional representation retains some meaningful intrinsic properties of the original data. This is done in order to improve the performance of models and algorithms, as well as to reduce the amount of data that needs to be processed. A GPA score is an example of a complex dataset of inputs being reduced to a single numerical representation.

Feature selection, where you decide what data variables or attributes are most important and which you can ignore. This process is foundational to dimensionality reduction and pattern recognition in all machine learning tasks. By focusing the model on a subset of features to use in further model construction, you can improve accuracy, save computation time and enhance model learning.

Here are the three main dimensionality reduction algorithms:

- **Principal Component Analysis (PCA)** — represents large datasets using a reduced, smaller set of “summary indices” or “principal components” that can be more easily visualized and analyzed. The technique is often used to emphasize variation and capture strong patterns in a dataset. This is also known as single value decomposition (SVD).
- **T-distributed Stochastic Neighbor Embedding (T-SNE)** — represents nonlinear dimensions by separating data that cannot be fitted to a straight line. It does so by mapping a probability distribution of all the inputs and introduces a new measurement known as perplexity, which is a user-defined parameter that sets the target number of neighbors for a central data point.

- **Uniform Manifold Approximation and Projection (UMAP)** — represents complex multi-dimensional spaces or geometric structures through the combining of discrete data points and line segments called simplices. When combined, these basic building blocks can provide fuzzy visualizations for 3D real-world objects.

Deep Learning

Deep learning models are neural networks that mimic the functioning of the human brain to find correlations and patterns by processing data with a specified logical structure. This filtering of data through three or more layers of processing allows training on labeled and unlabeled data simultaneously. Deep learning models are trained by using large sets of labeled data and neural network architectures that automate feature learning without the need for manual extraction. Unlike basic neural networks, deep learning model architectures consist of multiple hidden layers.

Here are the most common deep learning algorithm architectures:

- **Multi-layer Perceptron (MLP)** — a basic "feed-forward" neural network with node connections that only allow information to flow one way through the system, not backward or in a loop. It consists of one input layer, a hidden layer, and an output layer.
- **Convolution Neural Networks (CNN)** — inspired by the human visual cortex system, this feed-forward model is designed to process pixel data for image recognition and processing tasks. The convolution layer converts all pixels in its receptive field into single value.
- **Recurrent Neural Networks (RNN)** — the connections between the model's nodes form a directed or undirected graph (or loop) along a temporal sequence, allowing it to exhibit dynamic behavior in time and space. It's used to process time-series or sequential data, such as each consecutive word in a sentence.
- **Generative Adversarial Networks (GAN)** — taking a page from evolutionary processes in biological systems, this model is designed with two neural networks that compete in a zero-sum game of prediction performance in order to learn. Their combined output is new, "synthetic data" that's increasingly indistinguishable from the real-world data the model is training on.
- **Diffusion Models (DM)** — unlike GANs, which learn to map a random noisy image to a point in the training distribution, diffusion models progressively add irrelevant data known as *statistical noise* to an image. It then reverses this process, progressively removing the noise until it reveals an image that belongs to the training data's distribution.
- **Autoencoders** — this model imposes a bottleneck in the neural network that forces the encodings of compressed knowledge representations of the original inputs. It then learns

how to reconstruct an accurate representation of that data as an output. This form of reversible dimensionality reduction allows for more computationally expensive data processing.

- **Transformers** — this model learns language in context and thus the *semantic meaning* of information by tracking relationships in sequential data like the order of words in a sentence. It refers to this process of mapping word gender, plurality, and rules of grammar as *positional encoding* and utilizes a mathematical form of attention to understand how distant data points (or words) in a series can influence each other. Transformers are extremely powerful and require enormous amounts of data to be effective. One popular transformer model use case is helping monitor and anonymize data in pre-production systems, in real-time.

What Is the Best ML Model?

While there are many factors to evaluate, there is no one-size-fits-all answer to the question of which model to choose. It depends on the specific use case or task you are trying to accomplish, as well as on the number of features, associations, structures, and volume of the inputted data. Even when you do know these parameters, it is still recommended that you always start with the simplest model that can be applied to a particular problem and then gradually enhance its complexity.

How to Test ML Model Performance

Even if your model learns the training data well, in terms of accuracy, this does not mean it will be effective in predicting new information, since by design you forced the model to do this. The essential metric is how a model performs in new scenarios and environments–*in the wild*–beyond its training data. That's where the practice of model validation is key.

Model validation requires you to split your data into two — a training dataset, which generates the model, and a validation dataset to test the model’s prediction accuracy in novel situations. You can then make adjustments to improve accuracy by tuning your hyperparameters and cross-validating outcomes to help you determine whether a more complex model is needed. One of the best ways to improve the accuracy of a model is by training it on more data.

Sources of Training Data

Unfortunately, one of the biggest challenges engineers, software developers, and data scientists face is getting access to a supply of data that's sufficient to test an idea or design a new feature. This is because much of the data that exists today and that is used for training models contains sensitive or personally identifiable information (PII). Gaining access to such data faces ethical and legal hurdles that can be prohibitively costly or impossible to surmount. In our modern digital economy, this bottleneck stifles most research and innovation efforts being undertaken today.

However, as mentioned, recent innovations in advanced deep learning model architectures, such as GANs and DMs now enable the generation of safe, synthetic data that's as useful and in some cases even better than the real thing for training highly accurate AI/ML models.

Access and Build ML Training Data with Gretel

Gretel.ai offers on-demand software tools for identifying, labeling, and transforming existing sensitive datasets, and also a deep learning synthetic generation model for creating your own bespoke, high-quality data for training.

The more real-world possibilities you can feed your model, the better accuracy or generalization you will see for novel situations. That's the ultimate goal. So always be sure to train with either safe real-world datasets that are privacy-preserving or high-quality synthetic data that is properly generated and labeled.