Lesson Overview

In this lesson, our goal is to give you a high-level introduction to the field of machine learning, including the broader context in which this branch of computer science exists.

Here are the main topics we'll cover:

What machine learning is and why it's so important in today's world

The historical context of machine learning

The data science process

The types of data that machine learning deals with

The two main perspectives in ML: the statistical perspective and the computer science perspective

The essential tools needed for designing and training machine learning models

The basics of Azure ML

The distinction between models and algorithms

The basics of a linear regression model

The distinction between parametric vs. non-parametric functions

The distinction between classical machine learning vs. deep learning

The main approaches to machine learning

The trade-offs that come up when making decisions about how to design and training machine learning models

In the process, you will also train your first machine learning model using Azure Machine Learning Studio.

What is Machine Learning?

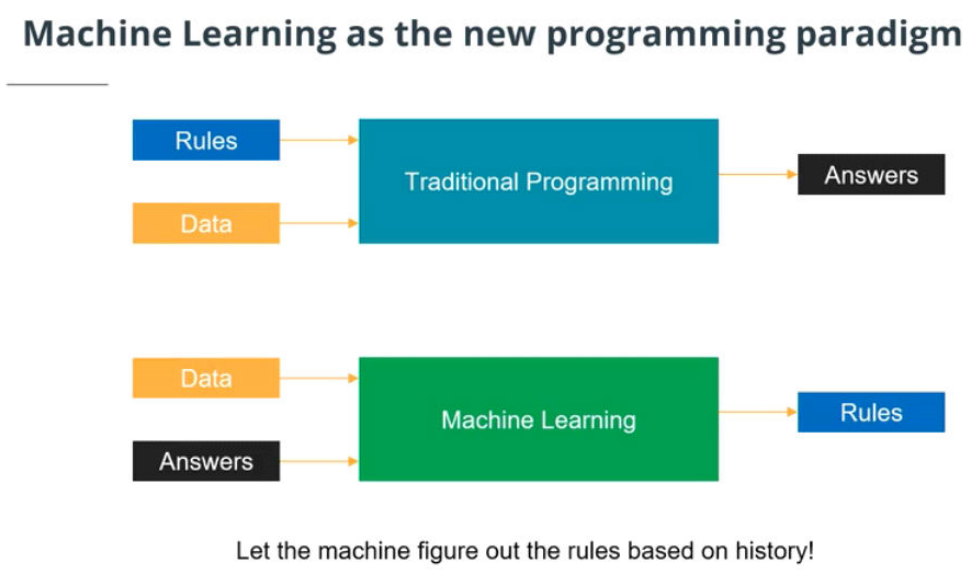
One of our goals in this lesson is to help you get a clearer, more specific understanding of what machine learning is and how it differs from other approaches.

Let's start with a classic definition. If you look up the term in a search engine, you might find something

like this:

Machine learning is a data science technique used to extract patterns from data, allowing computers to identify related data, and forecast future outcomes, behaviors, and trends.

Let's break that down a little. One important component of machine learning is that we are taking some data and using it to make predictions or identify important relationships. But looking for patterns in data is done in traditional data science as well. So how does machine learning differ?



Applications of Machine Learning

The applications of machine learning are extremely broad! And the opportunities cut across industry verticals. Whether the industry is healthcare, finance, manufacturing, retail, government, or education, there is enormous potential to apply machine learning to solve problems in more efficient and impactful ways.

We'll take a tour through some of the major areas where machine learning is applied, mainly just to give you an idea of the scope and type of problems that machine learning is commonly used to address.

Examples of Applied Machine Learning

Machine learning is used to solve an extremely diverse range of problems. For your reference, here are all the examples we discussed in the video, along with links to further reading in case you are curious and want to learn more about any of them:

Automate the recognition of disease

Trained physicians can only review and evaluate a limited volume of patients or patient images (X-rays, sonograms, etc.). Machine learning can be used to spot the disease, hence reducing physician burnout. For example, Google has trained a deep learning model to detect breast cancer and Stanford researchers have used deep learning models to diagnose skin cancer.

Recommend next best actions for individual care plans

With the mass digitization of patient data via systems that use EMRs (Electronic Medical Records) and EHRs (Electronic Health Records), machine learning can be used to help build effective individual care plans. For example, IBM Watson Oncology can help clinicians explore potential treatment options. More examples of how machine learning impacts healthcare can be found here.

Enable personalized, real-time banking experiences with chatbots

You've likely encountered this when you call a customer service number. Machine learning can be used to intercept and handle common, straightforward issues through chat and messaging services, so customers can quickly and independently resolve simple issues that would otherwise have required human intervention. With the chatbot, a customer can simply type in a question and the bot engages to surface the answer. Refer to this article to find more information about chatbot powered machine learning.

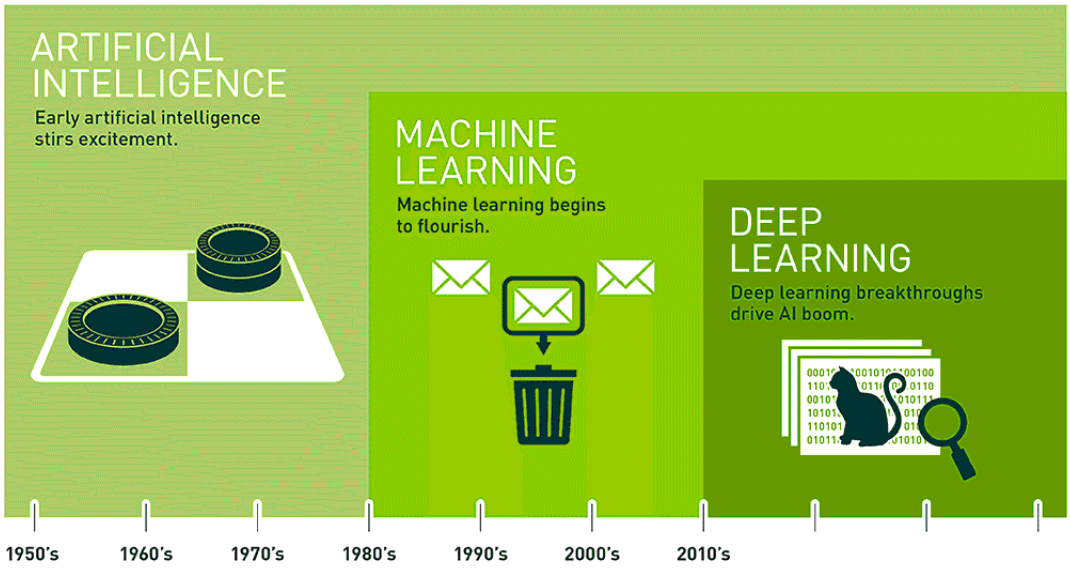
Identify the next best action for the customer

Real-time insights that incorporate machine learning tools—such as sentiment analysis—can help organizations assess the likelihood of a deal closing or the level of a customer's loyalty. Personally-tailored recommendations powered by machine learning can engage and delight customers with information and offers that are relevant to them.

Capture, prioritize, and route service requests to the correct employee, and improve response times

A busy government organization gets innumerable service requests on an annual basis. Machine learning tools can help to capture incoming service requests, to route them to the correct employee in real-time, to refine prioritization, and improve response times. Can check out this article if you're curious to learn more about ticket routing.

Brief History of Machine Learning



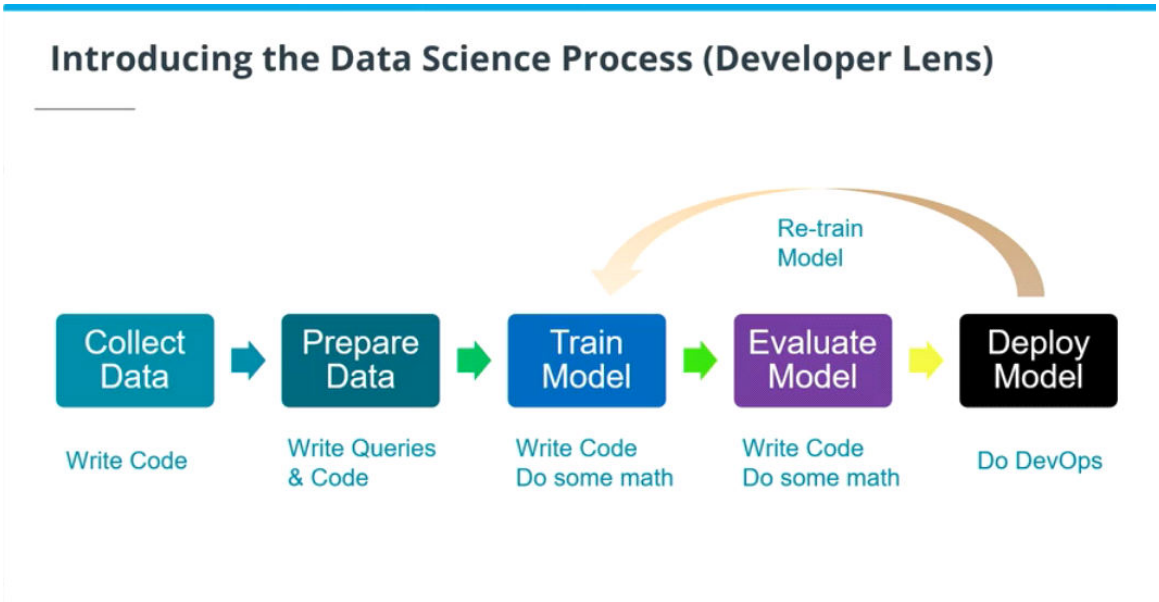
Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

The Data Science Process

Big data has become part of the lexicon of organizations worldwide, as more and more organizations look to leverage data to drive informed business decisions. With this evolution in business decision-making, the amount of raw data collected, along with the number and diversity of data sources, is growing at an astounding rate. This data presents enormous potential.

Raw data, however, is often noisy and unreliable and may contain missing values and outliers. Using such data for modeling can produce misleading results. For the data scientist, the ability to combine large, disparate data sets into a format more appropriate for analysis is an increasingly crucial skill.

The data science process typically starts with collecting and preparing the data before moving on to training, evaluating, and deploying a model. Let's have a look.



Common Types of Data

Numerical

Time - Series

Categorical

Text

Image

It's All Numerical in the End

Note that although we've described numerical data as a distinct category, it is actually involved in some way with all of the data types we've described. With the example of stock performance (above) the stock prices are numerical data points. So why do we give this as an example of "time-series data" rather than "numerical data"? It is the ordering of the numerical data points across points in time that leads us to call the data time-series data.

What is more, all data in machine learning eventually ends up being numerical, regardless of whether it is numerical in its original form, so it can be processed by machine learning algorithms.

For example, we may want to use gender information in the dataset to predict if an individual has heart disease. Before we can use this information with a machine learning algorithm, we need to transfer male vs. female into numbers, for instance, 1 means a person is male and 2 means a person is female, so it can

be processed. Note here that the value 1 or 2 does not carry any meaning.

Another example would be using pictures uploaded by customers to identify if they are satisfied with the service. Pictures are not initially in numerical form but they will need to be transformed into RGB values, a set of numerical values ranging from 0 to 255, to be processed.

Tabular Data

In machine learning, the most common type of data you'll encounter is tabular data—that is, data that is arranged in a data table. This is essentially the same format as you work with when you look at data in a spreadsheet.

Here's an example of tabular data showing some different clothing products and their properties:

SKU	Make	Color	Quantity	Price
908721	Guess	Blue	789	45.33
456552	Tillys	Red	244	22.91
789921	A&F	Green	387	25.92
872266	Guess	Blue	154	17.56

Notice how tabular data is arranged in rows and columns where each row describes a single product(e.g a shirt) while each column describes a property the products can have(e.g the colour of the product)

In general an item or entity is represented by a row, a property that the items or entities can have is represented by a column and a single value is represented by a cell.

Vectors

It is important to know that in machine learning we ultimately always work with numbers or specifically vectors.

A vector is simply an array of numbers, such as (1, 2, 3)—or a nested array that contains other arrays of numbers, such as (1, 2, (1, 2, 3)).

Vectors are used heavily in machine learning. If you have taken a basic course in linear algebra, then you are probably in good shape to begin learning about how they are used in machine learning. But if linear algebra and vectors are totally new to you, there are some great free resources available to help you learn. You may want to have a look at Khan Academy's excellent introduction to the topic here or check out Udacity's free Linear Algebra Refresher Course.

For now, the main points you need to be aware of are that:

All non-numerical data types (such as images, text, and categories) must eventually be represented as

numbers

In machine learning, the numerical representation will be in the form of an array of numbers—that is, a vector

As we go through this course, we'll look at some different ways to take non-numerical data and vectorize it (that is, transform it into vector form).

Scaling Data

Scaling data means transforming it so that the values fit within some range or scale, such as 0–100 or 0–1. There are a number of reasons why it is a good idea to scale your data before feeding it into a machine learning algorithm.

Let's consider an example. Imagine you have an image represented as a set of RGB values ranging from 0 to 255. We can scale the range of the values from 0–255 down to a range of 0–1. This scaling process will not affect the algorithm output since every value is scaled in the same way. But it can speed up the training process, because now the algorithm only needs to handle numbers less than or equal to 1.

Two common approaches to scaling data include standardization and normalization.

Standardization

Standardization rescales data so that it has a mean of 0 and a standard deviation of 1.

The formula for this is:

$$(x - \mu)/\sigma$$

We subtract the mean (μ) from each value (x) and then divide by the standard deviation (σ). To understand why this works, it helps to look at an example. Suppose that we have a sample that contains three data points with the following values: 50 100 150

The mean of our data would be 100, while the sample standard deviation would be 50.

Let's try standardizing each of these data points. The calculations are:

(50 − 100)/50 = -50/50 = -1

(100 − 100)/50 = 0/50 = 0

(150 − 100)/50 = 50/50 = 1

Thus, our transformed data points are:

-1 0 1

Again, the result of the standardization is that our data distribution now has a mean of 0 and a standard

deviation of 1.

Normalization

Normalization rescales the data into the range [0, 1].

The formula for this is:

$$(x - xmin) / (xmax - xmin)$$

For each individual value, you subtract the minimum value ($xmin$) for that input in the training dataset, and then divide by the range of the values in the training dataset. The range of the values is the difference between the maximum value ($xmax$) and the minimum value ($xmin$).

Let's try working through an example with those same three data points:

50 100 150

The minimum value ($xmin$) is 50, while the maximum value ($xmax$) is 150. The range of the values is $xmax - xmin$ = 150 − 50 = 100.

Plugging everything into the formula, we get:

(50 − 50)/100 = 0/100 = 0

(100 − 50)/100 = 50/100 = 0.5

(150 − 50)/100 = 100/100 = 1

Thus, our transformed data points are:

0 0.5 1

Again, the goal was to rescale our data into values ranging from 0 to 1—and as you can see, that's exactly what the formula did.

Encoding Categorical Data

As we've mentioned a few times now, machine learning algorithms need to have data in numerical form. Thus, when we have categorical data, we need to encode it in some way so that it is represented numerically.

There are two common approaches for encoding categorical data: ordinal encoding and one hot encoding.

Ordinal Encoding

In ordinal encoding, we simply convert the categorical data into integer codes ranging from 0 to (number

of categories – 1). Let's look again at our example table of clothing products:

SKU	Make	Color	Quantity	Price
908721	Guess	Blue	789	45.33
456552	Tillys	Red	244	22.91
789921	A&F	Green	387	25.92
872266	Guess	Blue	154	17.56

If we apply ordinal encoding to the Make property, we get the following:

Make	Encoding
A&F	0
Guess	1
Tillys	2

And if we apply it to the Color property, we get:

Color	Encoding
Red	0
Green	1
Blue	2

Using the above encoding, the transformed table is shown below:

SKU	Make	Color	Quantity	Price
908721	1	2	789	45.33
456552	2	0	244	22.91
789921	0	1	387	25.92
872266	1	2	154	17.56

One of the potential drawbacks to this approach is that it implicitly assumes an order across the categories. In the above example, Blue (which is encoded with a value of 2) seems to be more than Red (which is encoded with a value of 1), even though this is in fact not a meaningful way of comparing those values. This is not necessarily a problem, but it is a reason to be cautious in terms of how the encoded data is used.

One-Hot Encoding

One-hot encoding is a very different approach. In one-hot encoding, we transform each categorical value into a column. If there are n categorical values, n new columns are added. For example, the Color property has three categorical values: Red, Green, and Blue, so three new columns Red, Green, and Blue are added.

If an item belongs to a category, the column representing that category gets the value 1, and all other columns get the value 0. For example, item 908721 (first row in the table) has the color blue, so we put 1 into that Blue column for 908721 and 0 into the Red and Green columns. Item 456552 (second row in the table) has color red, so we put 1 into that Red column for 456552 and 0 into the Green and Blue columns.

If we do the same thing for the Make property, our table can be transformed as follows:

SKU	A&F	Guess	Tillys	Red	Green	Blue	Quantity	Price
908721	0	1	0	0	0	1	789	45.33
456552	0	0	1	1	0	0	244	22.91
789921	1	0	0	0	1	0	387	25.92
872266	0	1	0	0	0	1	154	17.56

One drawback of one-hot encoding is that it can potentially generate a very large number of columns.

Image Data

Images are another example of a data type that is commonly used as input in machine learning problems—but that isn't initially in numerical format. So, how do we represent an image as numbers? Let's have a look.

Taking a Closer Look at Image Data

Let's look a little closer at how an image can be encoded numerically. If you zoom in on an image far enough, you can see that it consists of small tiles, called pixels:

The color of each pixel is represented with a set of values:

In grayscale images, each pixel can be represented by a single number, which typically ranges from 0 to 255. This value determines how dark the pixel appears (e.g., 0 is black, while 255 is bright white).

In colored images, each pixel can be represented by a vector of three numbers (each ranging from 0 to 255) for the three primary color channels: red, green, and blue. These three red, green, and blue (RGB) values are used together to decide the color of that pixel. For example, purple might be represented as 128, 0, 128 (a mix of moderately intense red and blue, with no green).

The number of channels required to represent the color is known as the color depth or simply depth. With an RGB image, depth = 3, because there are three channels (Red, Green, and Blue). In contrast, a grayscale image has depth = 1, because there is only one channel.

Encoding an Image

Let's now talk about how we can use this data to encode an image. We need to know the following three things about an image to reproduce it:

Horizontal position of each pixel

Vertical position of each pixel

Color of each pixel

Thus, we can fully encode an image numerically by using a vector with three dimensions. The size of the vector required for any given image would be the height * width * depth of that image.

Other Preprocessing Steps

In addition to encoding an image numerically, we may also need to do some other preprocessing steps. Generally, we would want to ensure that the input images have a uniform aspect ratio (e.g., by making sure all of the input images are square in shape) and are normalized (e.g. subtract mean pixel value in a

channel from each pixel value in that channel). Some other preprocessing operations we might want to do to clean the input images include rotation, cropping, resizing, denoising, and centering the image.