***CHAPTER 1*. *The Machine Learning Landscape***

**Machine Learning is great for:**

* ***Problems for which existing solutions require a lot of fine-tuning or long lists of rules***: one Machine Learning algorithm can often simplify code and perform better than the traditional approach.
* ***Complex problems for which using a traditional approach yields no good solution***: the best Machine Learning techniques can perhaps find a solution.
* ***Fluctuating environments***: a Machine Learning system can adapt to new data.
* ***Getting insights about complex problems and large amounts of data***.

**Examples of Machine Learning Applications**

Some concrete examples of Machine Learning tasks, along with the techniques that can tackle them:

* ***Analysing images of products on a production line to automatically classify them*.**
  + this is **image classification**, typically performed using convolutional neural networks (CNNs; see Chapter 14).
* ***Detecting tumours in brain scans***
  + This is **semantic segmentation**, where each pixel in the image is classified (as we want to determine the exact location and shape of tumors), typically using CNNs as well.
* ***Automatically classifying news articles***
  + This is **natural language processing (NLP)**, and more specifically **text classification**, which can be tackled using recurrent neural networks (RNNs), CNNs, or Transformers (see Chapter 16).
* ***Automatically flagging offensive comments on discussion forums***
  + This is also **text classification**, using the same NLP tools.
* ***Summarizing long documents automatically***
  + This is a branch of NLP called **text summarization**, again using the same tools.
* ***Creating a chatbot or a personal assistant***
  + This involves many NLP components, including **natural language understanding (NLU)** and **question-answering modules**.
* ***Forecasting your company’s revenue next year, based on many performance metrics.***
  + This is a **regression task** (i.e., predicting values) that may be tackled using any regression model, such as a *Linear Regression* or *Polynomial Regression model* (see Chapter 4), a *regression SVM* (see Chapter 5), a *regression Random Forest* (see Chapter 7), or an *artificial neural network* (see Chapter 10). If you want to take into account sequences of past performance metrics, you may want to use RNNs, CNNs, or Transformers (see Chapters 15 and 16).
* ***Making your app react to voice commands.***
  + This is **speech recognition**, which requires processing audio samples: since they are long and complex sequences, they are typically processed using RNNs, CNNs, or Transformers (see Chapters 15 and 16).
* ***Detecting credit card fraud***
  + This is **anomaly detection** (see Chapter 9).
* ***Segmenting clients based on their purchases so that you can design a different marketing strategy for each segment.***
  + This is **clustering** (see Chapter 9).
* ***Representing a complex, high-dimensional dataset in a clear and insightful diagram***
  + This is **data visualization**, often involving **dimensionality reduction** techniques (see Chapter 8).
* ***Recommending a product that a client may be interested in, based on past purchases.***
  + This is a **recommender system**. One approach is to feed past purchases (and other information about the client) to an *artificial neural network* (see Chapter 10) and get it to output the most likely next purchase. This neural net would typically be trained on past sequences of purchases across all clients.
* ***Building an intelligent bot for a game***
  + This is often tackled using **Reinforcement Learning** (RL; see Chapter 18), which is a branch of Machine Learning that trains agents (such as bots) to pick the actions that will maximize their rewards over time (e.g., a bot may get a reward every time the player loses some life points), within a given environment (such as the game). The famous AlphaGo program that beat the world champion at the game of Go was built using RL.

**Types of Machine Learning Systems**

There are so many different types of Machine Learning systems that it is useful to classify them in broad categories, based on the following criteria[[1]](#footnote-1):

* **According to the amount and type of (human) supervision they get during training)**
  + Supervised Learning,
  + Unsupervised Learning,
  + Semisupervised Learning, and
  + Reinforcement Learning
* **Whether or not they can learn incrementally (on the fly) from a stream of incoming data**
  + Online Learning
  + Batch learning
* **How they generalize**
  + Whether they work by simply comparing new data points to known data points, or instead by detecting patterns in the training data and building a predictive model
    - Instance-based learning
    - Model-based learning

Let’s look at each of these criteria a bit more closely.

|  |
| --- |
| In Machine Learning an *attribute* is a data type (e.g., “mileage”), while a *feature* has several meanings, depending on the context, but generally means an attribute plus its value (e.g., “mileage = 15,000”). Many people use the words *attribute* and *feature* interchangeably. |

**Supervised learning**

Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output. Y = f(X). The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data. It is called supervised learning because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know the correct answers, the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance. The typical supervised learning tasks and some of the most important supervised learning algorithms:

* ***Classification task***
  + A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”.
* ***Regression task***
  + A regression problem is when the output variable is a numeric value, such as “dollars” or “weight”.
* ***Algorithms***
  + k-Nearest Neighbors
  + Linear Regression
  + Logistic Regression
  + Support Vector Machines (SVMs)
  + Decision Trees and Random Forests
  + Neural networks[[2]](#footnote-2)

**Unsupervised learning**

In unsupervised learning, as you might guess, the training data is unlabelled and the algorithms learn to inherent structure from the input data, somehow the system tries to learn without a teacher.

* **Clustering**
  + K-Means
  + DBSCAN
  + Hierarchical Cluster Analysis (HCA)
* **Anomaly detection and novelty detection**
  + ***Anomaly detection task*** — for example, detecting unusual credit card transactions to prevent fraud, catching manufacturing defects, or automatically removing outliers from a dataset before feeding it to another learning algorithm. The system is shown mostly normal instances during training, so it learns to recognize them; then, when it sees a new instance, it can tell whether it looks like a normal one or whether it is likely an anomaly.
  + A very similar task is ***novelty detection***: it aims to detect new instances that look different from all instances in the training set. This requires having a very “clean” training set, devoid of any instance that you would like the algorithm to detect. For example, if you have thousands of pictures of dogs, and 1% of these pictures represent Chihuahuas, then a novelty detection algorithm should not treat new pictures of Chihuahuas as novelties. On the other hand, anomaly detection algorithms may consider these dogs as so rare and so different from other dogs that they would likely classify them as anomalies.
  + ***Algorithms***
    - One-class SVM
    - Isolation Forest
* **Visualization and dimensionality reduction**
  + **Visualization** algorithms are also good examples of unsupervised learning algorithms: you feed them a lot of complex and unlabelled data, and they output a 2D or 3D representation of your data that can easily be plotted (Figure 1-9). These algorithms try to preserve as much structure as they can (e.g., trying to keep separate clusters in the input space from overlapping in the visualization) so that you can understand how the data is organized and perhaps identify unsuspected patterns.
  + A related **task is dimensionality reduction**, in which the goal is to simplify the data without losing too much information. One way to do this is to merge several correlated features into one. For example, a car’s mileage may be strongly correlated with its age, so the dimensionality reduction algorithm will merge them into one feature that represents the car’s wear and tear. This is called feature extraction.
  + ***Algorithms***
    - Principal Component Analysis (PCA)
    - Kernel PCA
    - Locally Linear Embedding (LLE)
    - t-Distributed Stochastic Neighbor Embedding (t-SNE)
* **Association rule learning**
  + The goal of *association rule learning task* is to dig into large amounts of data and discover interesting relations between attributes. For example, suppose you own a supermarket. Running an association rule on your sales logs may reveal that people who purchase barbecue sauce and potato chips also tend to buy steak. Thus, you may want to place these items close to one another.
  + ***Algorithms***
    - Apriori
    - Eclat

|  |
| --- |
| It is often a good idea to try to **reduce the dimension** of your training data using a dimensionality reduction algorithm before you feed it to another Machine Learning algorithm (such as a supervised learning algorithm). It will run much faster, the data will take up less disk and memory space, and in some cases it may also perform better. |

**Semisupervised learning**

Since labelling data is usually time-consuming and costly, you will often have large number of unlabeled instances and only some of the data is labelled. Some algorithms can deal with data that’s partially labeled. This is called *semisupervised learning.*

Some photo-hosting services, such as Google Photos, are good examples of this. Once you upload all your family photos to the service, it automatically recognizes that the same person A shows up in photos 1, 5, and 11, while another person B shows up in photos 2, 5, and 7. This is the unsupervised part of the algorithm (clustering). Now all the system needs is for you to tell it who these people are. Just add one label per person[[3]](#footnote-3) and it is able to name everyone in every photo, which is useful for searching photos.

Most semisupervised learning algorithms are combinations of unsupervised and supervised algorithms. For example

* D*eep belief networks* (DBNs) are based on unsupervised components called *restricted Boltzmann machines* (RBMs) stacked on top of one another. RBMs are trained sequentially in an unsupervised manner, and then the whole system is fine-tuned using supervised learning techniques.

**Reinforcement Learning**

The learning system, called an agent in this context, can observe the environment, select, and perform actions, and get rewards in return (or penalties in the form of negative rewards). It must then learn by itself what is the best strategy, called a policy, to get the most reward over time. A policy defines what action the agent should choose when it is in a given situation. For example, many robots implement Reinforcement Learning algorithms to learn how to walk.

**Batch learning**

In batch learning, the system is incapable of learning incrementally. If you want a batch learning system to know about new data

* you need to train a new version of the system from scratch on the full dataset (not just the new data, but also the old data)
* then stop the old system and launched (replace it with) the new one into production and runs without learning anymore; it just applies what it has learned.

This will generally take a lot of time and computing resources, so it is typically done offline, and it is called ***offline learning***. Fortunately, the whole process of training, evaluating, and launching a Machine Learning system can be automated easily.

**Disadvantages of Batch learning**

* Training on the full set of data requires a lot of computing resources (CPU, memory space, disk space, disk I/O, network I/O, etc.).
  + if your system needs to be able to learn autonomously and it has limited resources (e.g., a smartphone application or a rover on Mars) then carrying around large amounts of training data and taking up a lot of resources to train for hours every day is a showstopper.
  + If you have a lot of data and you automate your system to train from scratch every day, it will end up costing you a lot of money.
  + If the amount of data is huge, it may even be impossible to use a batch learning algorithm.
* If your system needs to adapt to rapidly changing data (e.g., to predict stock prices), then you need a more reactive solution.

**Online learning**

In online learning, you train the system incrementally by feeding it data instances sequentially, either individually or in small groups called ***mini-batches***. Each learning step is fast and cheap, so the system can learn about new data on the fly, as it arrives.

**Advantages**

* Online learning is great for systems that receive data as a continuous flow (e.g., stock prices) and need to adapt to change rapidly or autonomously.
* It is also a good option if you have limited computing resources: once an online learning system has learned about new data instances, it does not need them anymore, so you can discard them (unless you want to be able to roll back to a previous state and “replay” the data). This can save a huge amount of space.
* Online learning algorithms can also be used to train systems on huge datasets that cannot fit in one machine’s main memory (this is called *out-of-core* learning).
  + The algorithm loads part of the data, runs a training step on that data, and repeats the process until it has run on all the data.

**Disadvantages**

* A big challenge with online learning is that if bad data is fed to the system, the system’s performance will gradually decline. If it’s a live system, your clients will notice.
  + For example, bad data could come from a malfunctioning sensor on a robot, or from someone spamming a search engine to try to rank high in search results.
  + **To reduce this risk**, you need to monitor your system closely and promptly switch learning off (and possibly revert to a previously working state) if you detect a drop in performance. You may also want to monitor the input data and react to abnormal data (e.g., using an anomaly detection algorithm).

**Instance-based learning**

Instance-based learning system learns the examples by heart, then generalizes to new cases by using a similarity measure to compare them to the learned examples (or a subset of them).



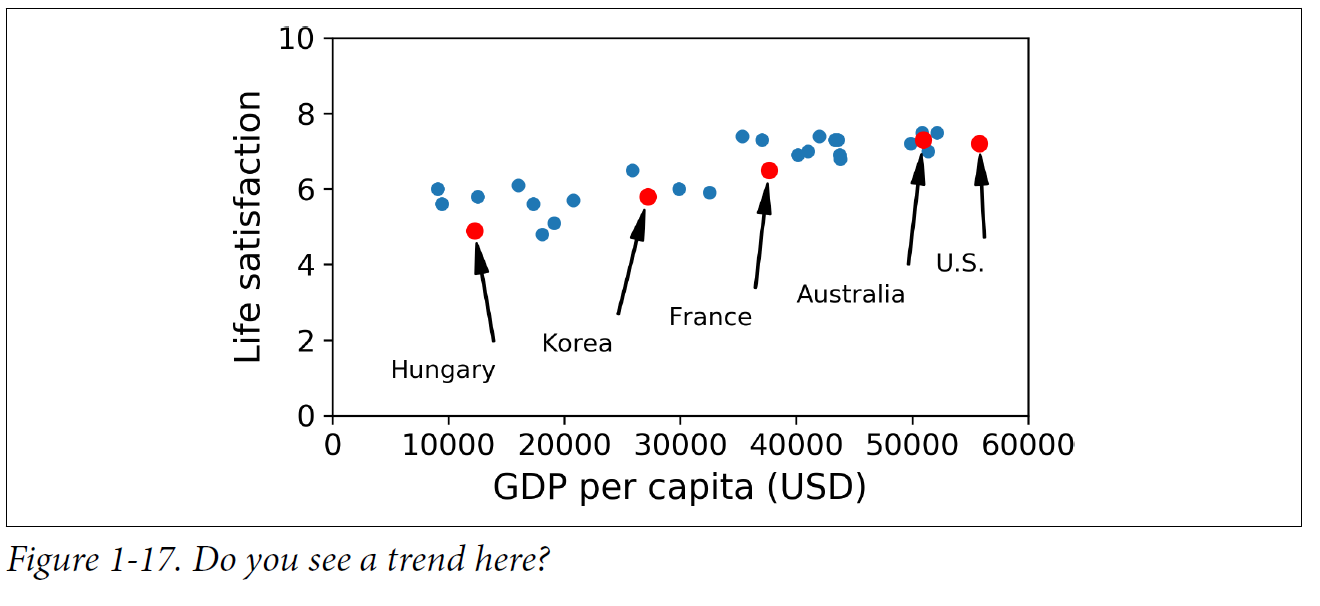
**Model-based learning**

Another way to generalize from a set of examples is to build a model (based on - defining some parameters) of these examples and then use that model to make predictions. This is called ***model-based learning***. Before using the model, we need to specify a performance measure. We can either define a *utility function (or fitness function)* that measures how good your model is, or we can define a *cost function* that measures how bad it is.

For *Linear Regression problems*, people typically use a cost function that measures the distance between the linear model’s predictions and the training examples; the objective is to minimize this distance.

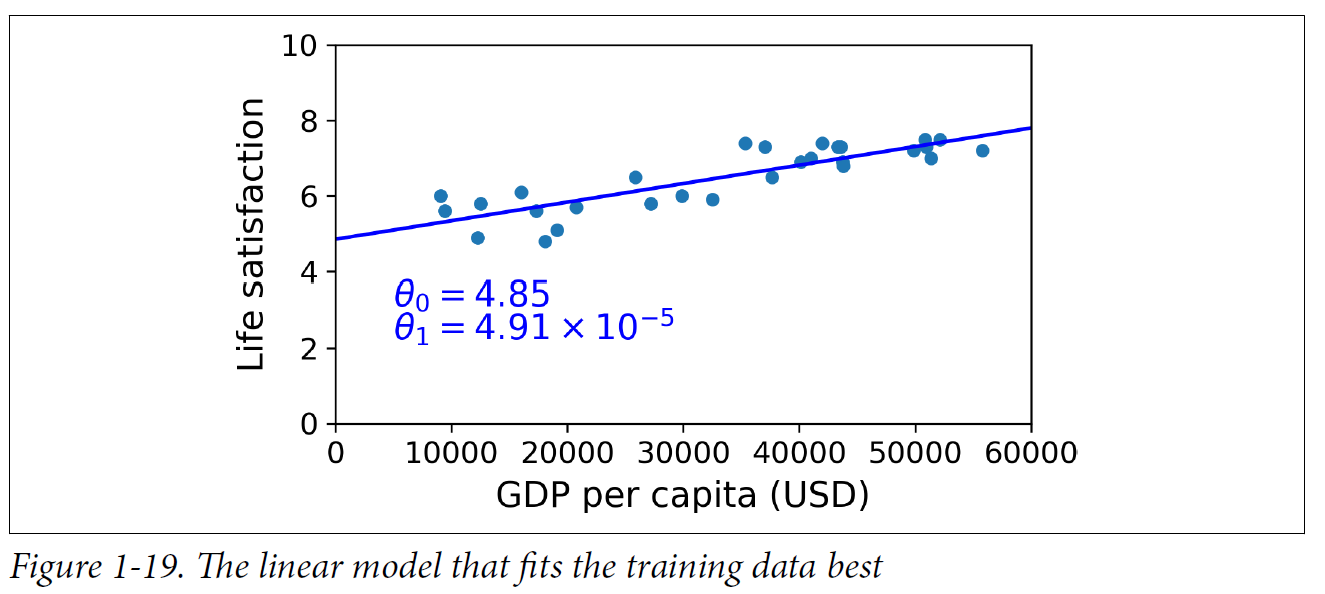
**Example:** suppose you want to know if money makes people happy, so you download the Better Life Index data from the OECD’s website and stats about gross domestic product (GDP) per capita from the IMF’s website. Then you join the tables and sort by GDP per capita.

Let’s plot the data for these countries.



There does seem to be a trend here! Although the data is noisy (i.e., partly random), it looks like life satisfaction goes up more or less linearly as the country’s GDP per capita increases. So, you decide to model life satisfaction as a linear function of GDP per capita. This step is called model selection: you selected a linear model of life satisfaction with just one attribute, GDP per capita.

You feed it your training examples, and it finds the parameters that make the linear model fit best to your data. This is called *training* the model. In our case, the algorithm finds that the optimal parameter values are ***θ*0** = 4.85 and ***θ*1** = 4.91 × 10–5.



You are finally ready to run the model to make predictions.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Machine  Learning | **Learning Type** | Tasks |  |  |
| Supervised | *Classification* | A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”. |  |
| *Regression* | A regression problem is when the output variable is a numeric value, such as “dollars” or “weight”. |  |
| Unsupervised | Clustering |  |  |
| Anomaly detection & Novelty detection |  |  |
| Visualization & Dimensionality reduction |  |  |
| Association rule learning |  |  |
| Semisupervised |  |  |  |
| Reinforcement |  |  |  |

**Exercises**

In this chapter we have covered some of the most important concepts in Machine Learning. In the next chapters we will dive deeper and write more code, but before we do, make sure you know how to answer the following questions:

1. How would you define Machine Learning?
2. Can you name four types of problems where it shines?
3. What is a labelled training set?
4. What are the two most common supervised tasks?
5. Can you name four common unsupervised tasks?
6. What type of Machine Learning algorithm would you use to allow a robot to walk in various unknown terrains?
7. What type of algorithm would you use to segment your customers into multiple groups?
8. Would you frame the problem of spam detection as a supervised learning problem or an unsupervised learning problem?
9. What is an online learning system?
10. What is out-of-core learning?
11. What type of learning algorithm relies on a similarity measure to make predictions?
12. What is the difference between a model parameter and a learning algorithm’s hyperparameter?
13. What do model-based learning algorithms search for? What is the most common strategy they use to succeed? How do they make predictions?
14. Can you name four of the main challenges in Machine Learning?
15. If your model performs great on the training data but generalizes poorly to new instances, what is happening? Can you name three possible solutions?
16. What is a test set, and why would you want to use it?
17. What is the purpose of a validation set?
18. What is the train-dev set, when do you need it, and how do you use it?
19. What can go wrong if you tune hyperparameters using the test set?

1. These criteria are not exclusive; you can combine them in any way you like. For example, a state-of-the-art spam filter may learn on the fly using a deep neural network model trained using examples of spam and ham; this makes it an online, model based, supervised learning system. [↑](#footnote-ref-1)
2. Some neural network architectures can be unsupervised, such as autoencoders and restricted Boltzmann machines. They can also be semisupervised, such as in deep belief networks and unsupervised pretraining. [↑](#footnote-ref-2)
3. That’s when the system works perfectly. In practice it often creates a few clusters per person, and sometimes mixes up two people who look alike, so you may need to provide a few labels per person and manually clean up some clusters. [↑](#footnote-ref-3)