Def Data mining: Applying ML techniques to dig into large amounts of data can help discover patterns that were not immediately apparent.

## To summarize, 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.

* 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.

# Types of Machine Learning

Machine Learning systems can be classified according to the amount and type of supervision they get during training. There are four major categories: supervised learning, unsupervised learning, semisupervised learning, and Reinforcement Learning

## Supervised Learning

Def Supervised Learning: In supervised learning, the training set you feed to the algorithm includes the desired solutions, called labels

A typical supervised learning task is classification. The spam filter is a good example of this: it is trained with many examples emails along with their class (spam or ham), and it must learn how to classify new emails.

Another typical task is to predict a target numeric value

## UNSUPERVISED LEARNING

Def unsupervised learning: as you might guess, the training data is unlabeled ([Figure 1-7](https://learning.oreilly.com/library/view/hands-on-machine-learning/9781492032632/ch01.html#unsupervised_learning_diagram)). The system tries to learn without a teacher.

Def dimensionality reduction, in which the goal is to simplify the data without losing too much information

Def feature extraction: 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

TIP: 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

Another task of unsupervised learning is anomaly detection in which 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.

Another common unsupervised task is association rule learning, in which the goal 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.

## SEMISUPERVISED LEARNING

DEF Semi-supervised learning: Since labeling data is usually time-consuming and costly, you will often have plenty of unlabeled instances, and few labeled instances. Some algorithms can deal with data that’s partially labeled, this is semi-supervised learning.

Most semisupervised learning algorithms are combinations of unsupervised and supervised algorithms. Google Photos, is a 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 person4 and it is able to name everyone in every photo, which is useful for searching photos.

Def Reinforcement Learning: In 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.

# Batch and Online Learning

Another criterion used to classify Machine Learning systems is whether or not the system can learn incrementally from a stream of incoming data.

## BATCH LEARNING

In batch learning, the system is incapable of learning incrementally: it must be trained using all the available data. This will generally take a lot of time and computing resources, so it is typically done offline. First the system is trained, and then it is launched into production and runs without learning anymore; it just applies what it has learned. This is called offline learning. 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

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 frees up space as data can be deleted.

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)

One important parameter of online learning systems is how fast they should adapt to changing data: this is called the learning rate. If you set a high learning rate, then your system will rapidly adapt to new data, but it will also tend to quickly forget the old data (you don’t want a spam filter to flag only the latest kinds of spam it was shown).

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. You may also want to monitor the input data and react to abnormal data (e.g., using an anomaly detection algorithm).

# Instance-Based Versus Model-Based Learning

more way to categorize Machine Learning systems is by how they generalize

## INSTANCE-BASED LEARNING

DEF instance-based learning: The system learns the examples by heart, then generalizes to new cases by using a similarity measure to compare them to the learned examples.

## MODEL-BASED LEARNING

DEF model-based learning: To generalize from a set of examples is to build a model of these examples and then use that model to make predictions

Def Noise: Randomness

Def Utility function (or fitness function): Measures how good your model is.

Def Cost function: Measures how bad you model is.

# Main Challenges of Machine Learning

## Insufficient Quantity of Training Data

It takes a lot of data for most Machine Learning algorithms to work properly. Even for very simple problems you typically need thousands of examples, and for complex problems such as image or speech recognition you may need millions of examples

## Nonrepresentative Training Data

In order to generalize well, it is crucial that your training data be representative of the new cases you want to generalize to. This is true whether you use instance-based learning or model-based learning. It is crucial to use a training set that is representative of the cases you want to generalize to. This is often harder than it sounds: if the sample is too small, you will have sampling noise (i.e., nonrepresentative data as a result of chance), but even very large samples can be nonrepresentative if the sampling method is flawed. This is called sampling bias(A poll of California voters as the “voice” of the country).

## Poor Quality Data

If your training data is full of errors, outliers, and noise (e.g., due to poor-quality measurements), it will make it harder for the system to detect the underlying patterns, so your system is less likely to perform well.

If some instances are clearly outliers, it may help to simply discard them or try to fix the errors manually.

If some instances are missing a few features (e.g., 5% of your customers did not specify their age), you must decide whether you want to ignore this attribute altogether, ignore these instances, fill in the missing values (e.g., with the median age), or train one model with the feature and one model without it.

## Irrelevant Features

Your system will only be capable of learning if the training data contains enough relevant features and not too many irrelevant ones. A critical part of the success of a Machine Learning project is coming up with a good set of features to train on. This process, called feature engineering, involves the following steps:

1. Feature selection (selecting the most useful features to train on among existing features)

2. Feature extraction (combining existing features to produce a more useful one⁠—as we saw earlier, dimensionality reduction algorithms can help)

3. Creating new features by gathering new data

## Overfitting the training data

Def Overfitting: it means that the model performs well on the training data, but it does not generalize well.

Overfitting happens when the model is too complex relative to the amount and noisiness of the training data. Here are possible solutions:

1. Simplify the model by selecting one with fewer parameters (e.g., a linear model rather than a high-degree polynomial model), by reducing the number of attributes in the training data, or by constraining the model.

2. Gather more training data.

3. Reduce the noise in the training data (e.g., fix data errors and remove outliers).

Def regularization: Constraining a model to make it simpler and reduce the risk of overfitting.

Def hyperparameter: The amount of regularization to apply during learning can be controlled. it is not affected by the learning algorithm itself; it must be set prior to training and remains constant during training,

## Underfitting the training data

Def Underfitting: Is the opposite of overfitting: it occurs when your model is too simple to learn the underlying structure of the data.

For example, a linear model of life satisfaction is prone to underfit; reality is just more complex than the model, so its predictions are bound to be inaccurate, even on the training examples.

Here are the main options for fixing this problem:

1. Select a more powerful model, with more parameters.

2. Feed better features to the learning algorithm (feature engineering).

3. Reduce the constraints on the model (e.g., reduce the regularization hyperparameter).

Tuning

Def validation set: Is used to compare models. It makes it possible to select the best model and tune the hyperparameters.

Question: What is the train-dev set, when do you need it, and how do you use it?

A: The train-dev set is used when there is a risk of mismatch between the training data and the data used in the validation and test datasets (which should always be as close as possible to the data used once the model is in production). The train-dev set is a part of the training set that’s held out (the model is not trained on it). The model is trained on the rest of the training set, and evaluated on both the train-dev set and the validation set. If the model performs well on the training set but not on the train-dev set, then the model is likely overfitting the training set. If it performs well on both the training set and the train-dev set, but not on the validation set, then there is probably a significant data mismatch between the training data and the validation + test data, and you should try to improve the training data to make it look more like the validation + test data.

Question: What can go wrong if you tune hyperparameters using the test set?

A: If you tune hyperparameters using the test set, you risk overfitting the test set, and the generalization error you measure will be optimistic (you may launch a model that performs worse than you expect).