Introduction to Machine Learning

Taras Rashkevych

December 21, 2021

Summary

Abstract

My personal notes on the contents of the Introduction to Machine Learning course held by professor Elisa Ricci at the University of Trento. These notes should provide a broad and complete introduction to the world of Machine Learning and Statistical Pattern Recognition and also be the basis for further courses on more deep topics in this area.

1 Introduction

1.1 Overview

The following are the three main families of machine learning methods, which are also the topics of these notes:

- ***Supervised Learning***: parametric/non-parametric algorithms(e.g. nearest neighbors, decision trees and random forests), kernel methods, deep neural networks (e.g. feedforward, convolutional and recurrent networks).
- ***Unsupervised Learning***: clustering, dimensionality reduction, autoencoders, deep generative models.
- ***Reinforcement Learning***: these notes only cover a high-level introduction.

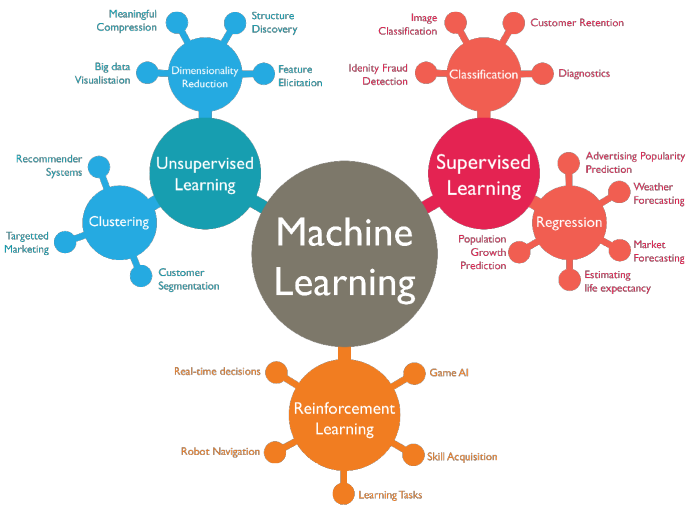


Figure 1: High-level description of the world of Machine Learning

1.2 What is Machine Learning?

There are several definitions of what Machine Learning is, among which we can find the following that perfectly reflect its conceptual nature:

> *"Machine learning is the study of computer algorithms that improve automatically through experience. It is seen as a part of artificial intelligence."*
>
> — Wikipedia

> *"Machine learning is the science of getting computers to act without being explicitly programmed."*
>
> — A. Samuel (1959)

Given all these expressive and equivalent definitions, the general idea of what Machine Learning is and how it differentiates from the traditional way of programming can be summarized by the following image:

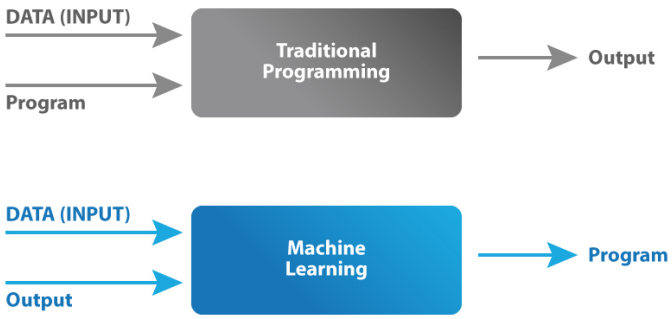


Figure 2: Machine Learning vs Traditional Programming

By observing the image it becomes clear that in traditional programming the programmer writes a program and then gives it some input data in order to produce some output. In machine learning, on the other hand, the conceptual model is totally different because in this case the programmer still gives some input data but also gives some known result (e.g. data with annotations/labels), which in combination with the previous ones are used to produce a program. In this way the machine learns from the input data and the data with annotations to be able to write a program.

Some problems are so hard to solve that it becomes not impossible but extremely difficult to write programs that solve them and even if such programs are produced by writing them manually then it is likely that these programs will not be able to provide sufficiently satisfiable results. So in these cases it is preferable to make the machine learn from the data and write the program that will provide some better results. An excellent example is a robot that has to learn how to walk because it is extremely difficult to write a suitable program for this kind of task but it is extremely useful to make the robot learn from the data because in this way it will also capture the real-world noise and in general all the unexpected things, which could not be modelled by a program written manually. Unfortunately there are many other examples in which the machine learning approach seems to be the most promising. Nevertheless all the solutions obtained with the machine learning approach present a common structure, in the sense that there are always *three fundamental elements* which are the following:

1. ***Data***: *a lot of data* is required in order to build complex and more or less reliable models.

2. ***Algorithms***: there are *a lot of algorithms* that can process the data mentioned in point 1, each of which has its own peculiarities. The choice of a particular algorithm always depends on the task to be addressed.

3. ***Model***: this is the result of applying the algorithms mentioned in point 2 to the data mentioned in point 1, which will be used on future data *similar* to the ones that were used for its training. So this result is somehow the summary of the knowledge that has been extrapolated from the data and represents the so-called *"knowledge"* that computers acquire by processing the data through the algorithms.

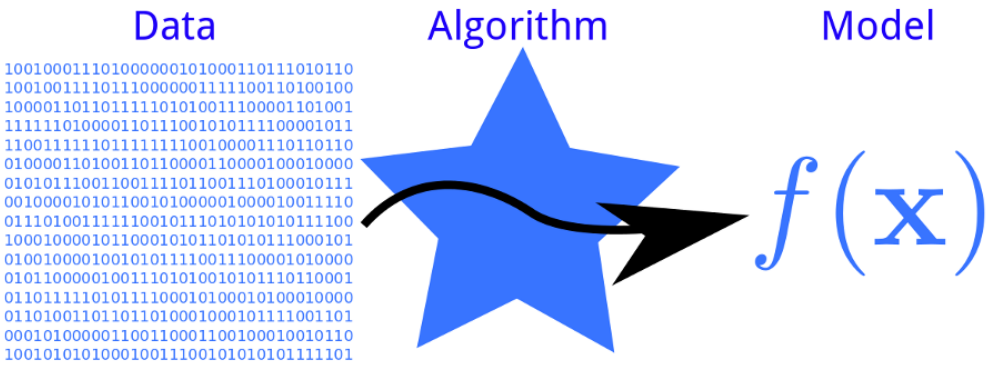


Figure 3: Three fundamental elements in Machine Learning

So the main reason why machine learning is so important and so widely used consists in the **Hardness** of manually writing programs that should represent satisfiable solutions to a particular class of problems. This hardness is then instantiated based on the particular nature of the problem:

- **Lack of Expertise**: Human beings have no idea how they should program the robot to navigate the surface of Mars because until now no human being has had the opportunity to explore this planet directly. So it is better for the machine to learn what to do autonomously.

- **Lack of Expressiveness**: Sometimes it is too difficult to explain the human experience such as sight or hearing and so to define a precise set of rules to follow. A great example is speech recognition where deducing the person's name from a particular waveform becomes significantly complex, therefore the best solution is to learn from the data and improve performance accordingly.

- **Personalization**: There are situations in which some kind of personalization is required and is not feasible for a manually written program to provide such personalization to all customers. Excellent examples are personalized medicine and the recommendations provided by streaming services.

- **Big Data**: In almost all situations in which machine learning is used a big amount of data is required for model training, but there are some situations in which the amount of data to be processed and to be reasoned about in order to achieve a particular goal is exceptionally large and therefore could not be handled by a manually written program. A great example is genomics where the amount of data to be used is extremely large.

It should be mentioned though that there are situations in which machine learning **is not useful.** Generally this happens when the rule, which determines how the machine is supposed to act, is perfectly known. So if the final goal is to calculate a payroll or perform a calculation using a well-known physical law, then traditional programming is perfect here.

After having seen the main reasons why machine learning is so widely used, it is mandatory to mention some of its most important practical applications:

- **Pattern Recognition**: This classic application consists in recognizing patters, which means that, given a certain input such as an image of handwritten digits, facial identities, facial expressions or a medical image, it is possible to recognize certain similarities and continuous repetitions of some structures seen before and be able to deduce the targeted content of that input. It is useful to mention that images are not the only available data type used in pattern recognition.

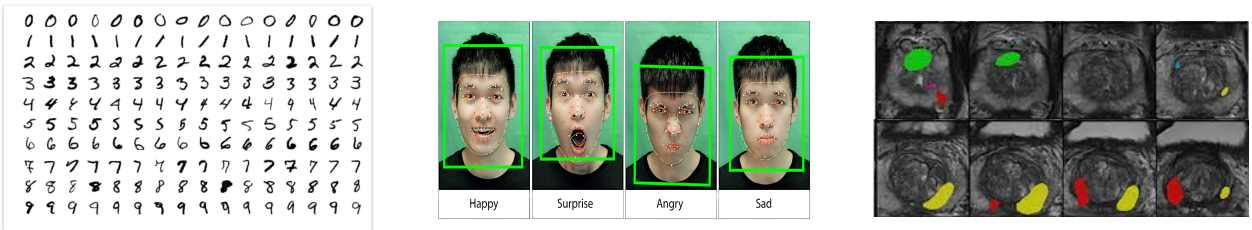


Figure 4: Some examples of Pattern Recognition

- **Pattern Generation**: This is another application which is extremely used nowadays and consists in generating patterns, which means that, given a certain distribution of data to learn from, it is possible to produce samples based on these data. For example with this technique it is possible to generate fake images and artificial motion sequences.

Figure 5: Some generated portraits of famous people

- **Anomaly Detection**: This is another important application which consists in detecting anomalies, which means that, given a certain amount of data, it is possible to produce machine learning models that will be able to predict unusual behavior. Some relevant examples are unusual credit card transactions, unusual patterns of sensor readings, unusual video surveillance frames and unusual system logins.

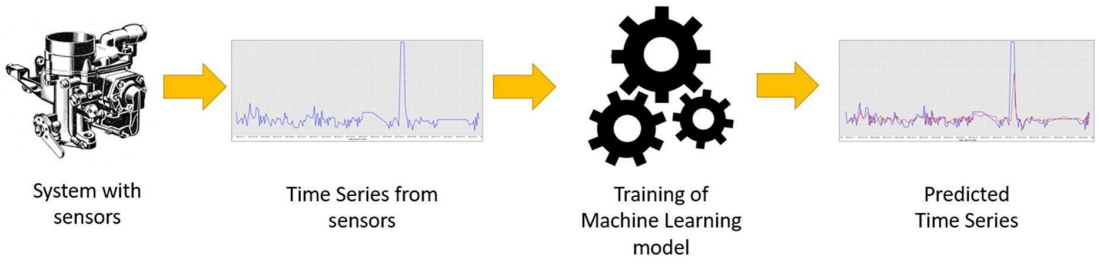
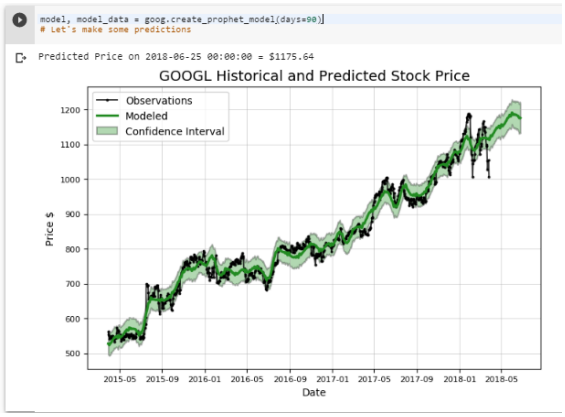


Figure 6: Anomaly detection in a system with sensors

- **Prediction**: This is an extremely important application in finance since it provides predictions on future stock prices or currency exchange rates. Nowadays it is also used in autonomous driving to predict future moves of people and other vehicles and to avoid possible accidents. Moreover, it is also used in gaming to predict future best moves, as it has been demonstrated by the AlphaGo program developed by the Google DeepMind group.



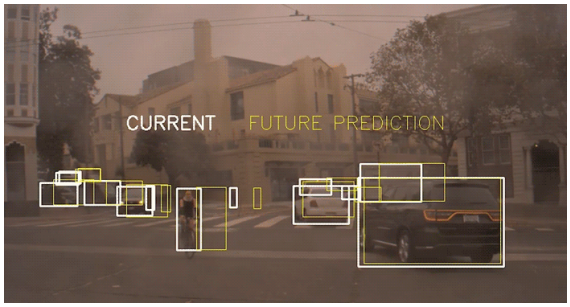


Figure 8: Some examples of Prediction

At this point, it is worth mentioning some other definitions of what Machine Learning is, provided by some big experts in this field:

> *"It is concerned with the automatic discovery of regularities in data through the use of computer algorithms and with the use of these regularities to take actions."*
>
> — Christopher M. Bishop

"The goal of machine learning is to develop methods that can automatically detect patterns in data, and then to use the uncovered patterns to predict future data or other outcomes of interest."

— Kevin P. Murphy

"Machine learning is about predicting the future based on the past."

— Hal Daume III

Given these last three definitions, it is possible to detect two fundamental steps of machine learning consisting in automatically discovering some regularities(i.e. patterns) and applying these regularities to take some actions, such as predicting future data. At this point, it becomes natural to introduce the typical machine learning pipeline:

1. ***Training***: At this stage of the process the past knowledge is represented by the so-called *training data*, which will be passed through a particular *algorithm* to produce the so-called *model*, also called *predictor*. During this stage, the so-called *feature extraction* also takes place, which basically consists in extracting from the raw data of the training set some kind of representations, called *features*, which are essential for adequate processing. These features will also be used by the aforementioned model to process future data.

2. ***Testing(Inference)***: At this stage, on the other hand, the first component of the future knowledge is represented by the so-called *testing data*, which will be passed to the *model*, generated during the first stage, to produce some *predictions*, which are the second component of the future knowledge.

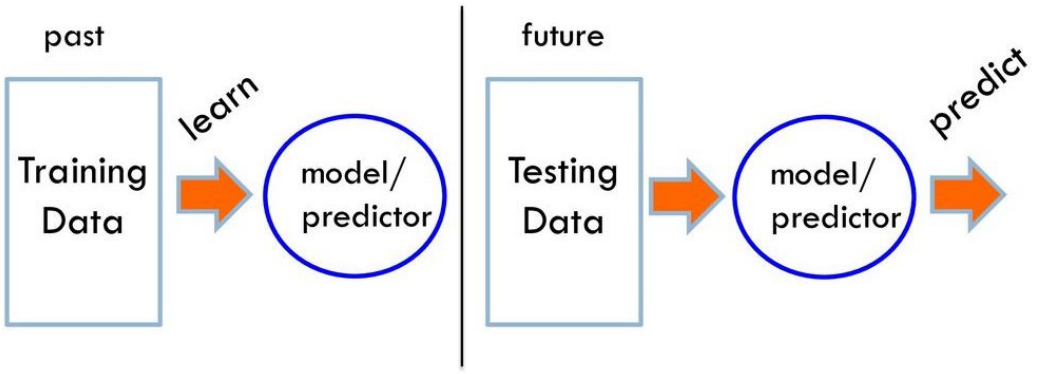


Figure 9: Typical machine learning process

The last definition of Machine Learning, which will be seen in this section and which introduces the concept of performance measure, is the following:

> *"A computer program is said to learn from* ***experience E*** *with respect to some class of* ***tasks T*** *and* ***performance measure P****, if its performance at tasks in T, as measured by P, improves with experience E."*
>
> — T. Mitchell (1970)

Basically what this definition states is that machine learning is the study of algorithms that produce models that:

- improve their performance P
- at some task T
- with experience E

At this point, thanks to this definition, it is possible to describe any machine learning problem by defining the triplet $\langle T, P, E \rangle$, which is then instantiated based on the particular nature of the problem:

- **Example 1.1** *T: Recognizing handwritten words. P: Percentage of words correctly classified. E: Database of human-labeled images of handwritten words.*
- **Example 1.2** *T: Driving on four-lane highways using vision sensors. P: Average distance traveled before a human-judged error. E: A sequence of images and steering commands recorded while observing a human driver.*
- **Example 1.3** *T: Categorize email messages as spam or legitimate. P: Percentage of email messages correctly classified. E: Database of emails, some with human-given labels.*

Given all these high-level definitions, reasons of usage, practical applications and some more detailed explanations of the fundamental elements of machine learning, it is worth mentioning some of the real-world success stories:

- *Face Detection*: The first working system in 2002.
- *Pedestrian Detection*: The first working system in 2005.
- *Body Tracking(RGB-D)*: Used for example in gaming.

1.3 What is Deep Learning?

In order to understand what Deep Learning is, it should be useful to observe the image below that represents in chronological order the popularity growth of Artificial Intelligence and its two most famous subsets, which are Machine Learning and Deep Learning.

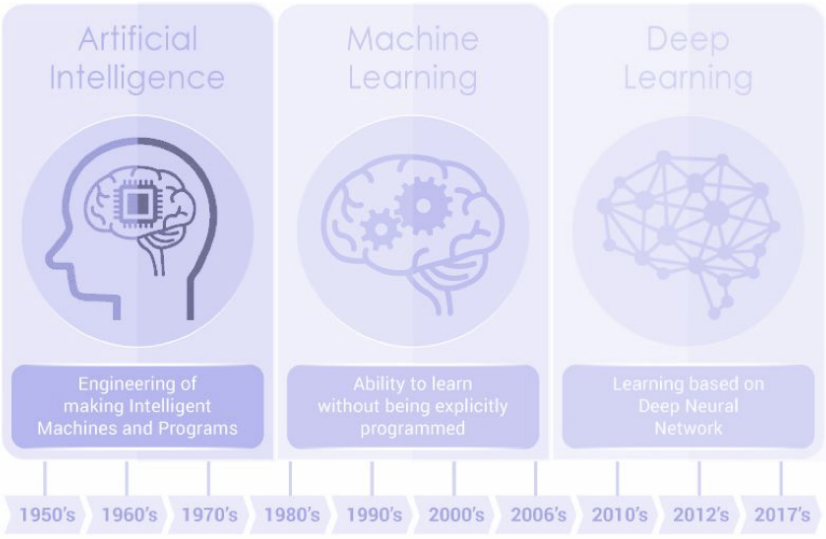


Figure 10: Artificial Intelligence, Machine Learning and Deep Learning

It immediately becomes clear that deep learning is now the most studied field(even if the idea of neural networks on its own is quite old) of the entire artificial intelligence area and is also the most promising one. In fact, most systems nowadays that apply some kind of machine learning are actually based on deep learning models. Furthermore, to understand even more deeply the relationships between the three areas mentioned above, the next image could be useful:

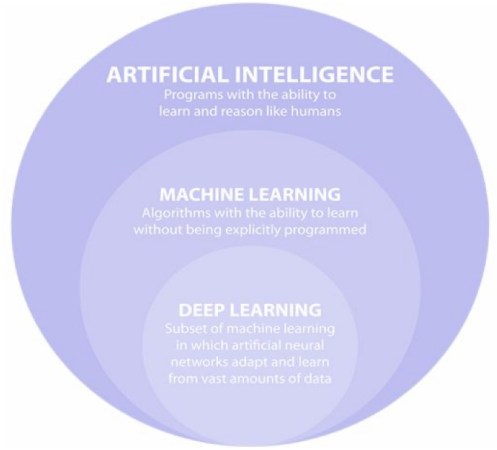


Figure 11: Artificial Intelligence, Machine Learning and Deep Learning

So, as the above images suggest, deep learning is the most recent actually working accomplishment of machine learning and artificial intelligence in general, which allows constructing complex and efficient machine learning models that are based on the so-called artificial neural networks, which adapt and learn from particularly large amounts of data. These neural networks actually consist of several layers of nodes between input and output that apply, instead of using the feature extraction process, a hierarchical processing so that the network itself automatically learns a mapping between the initial raw data and the final output. In fact, in order to produce this mapping, some form of representation of the input data is computed on each individual layer, with the abstraction gradually increasing(i.e. from particular details to general concepts) by aggregating the information from the lowest layers near the input layer towards the highest layers near the output layer. A summary of this process can be seen in the image below:

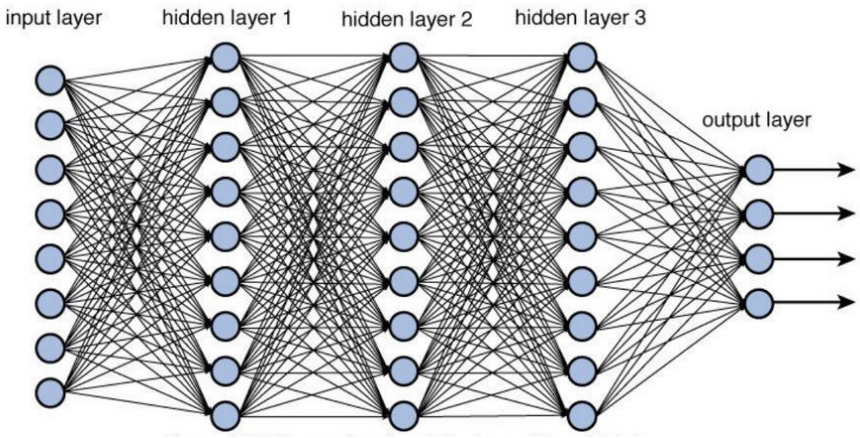


Figure 12: Deep Neural Network

At this point, it is worth mentioning some definitions of artificial intelligence and deep learning, provided by big experts in these areas:

> *"Our ultimate objective is to make programs that learn from their experience as effectively as humans do."*
>
> — J. McCarthy(1958)

> *"Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction"*
>
> — G. Hinton — Y. LeCun — Y. Bengio

The actual article about deep learning, its fundamental idea, and its applications written by the godfathers of deep learning(they won the 2018 Turing Award for conceptual and engineering breakthroughs that have made deep neural networks a critical component of computing) can be found at this address: `https://www.nature.com/articles/nature14539`.

At this point it is useful to mention some extremely important real-world success stories that made deep learning so popular and so widely used:

- *Object Recognition*: This is a general machine learning task that is composed of other two machine learning tasks called Object Detection and Object Classification. The former consists in assigning the position of that particular object by indicating the so-called bounding box, which defines the rectangular boundaries of where the object is located. The latter consists in assigning to each object present in an image or in a frame of a video a label indicating the category to which that particular object belongs and a parameter indicating the degree of accuracy. The first actually working deep learning system that solves this general task on the dataset called ImageNet was created in 2012. Furthermore, in 2015, the error rate of another system on the same dataset has decreased to the level of a human being, which once again underlines the power of deep learning.

- *Image Captioning*: This is a natural extension of Image Classification and a much more challenging task, which consists in assigning a textual description to an image or to a frame of a video. The first actually working deep learning system that solves this kind of task was created in 2015 and an excellent example of a similar system can be seen in this youtube video. Nowadays the systems that solve this kind of task have been further improved in performance.

- *Image Synthesis*: This is another machine learning task that consists in automatically generating plausible images associated with the sketch of a scene passed as input. Such a system has been successfully developed by Nvidia.

- *Speech Recognition*: This is a self-explanatory machine learning task that has been studied for years and has been significantly improved in performance by using some deep learning techniques starting from 2009.

- *Neural Machine Translation*: This is a machine learning task that consists in translating sentences from one language to another and that has also been significantly improved in performance by using some deep learning approaches starting from 2014.

- *Real-Time Voice Translation*: This is a machine learning task that consists in translating a person's speech in real-time by generating an artificial voice that speaks in the target language. An excellent example of a system that performs this kind of task can be seen in this youtube video.

- *AlphaGO*: This is the aforementioned program that has been developed by the Google DeepMind group with the purpose of predicting the best future moves in an ancient Chinese game called Go and that has actually beaten the best world player at Go. An interesting fact is that a movie has been made about this story and the trailer can be seen in this youtube video.

To conclude this subsection it is mandatory to mention the main reasons why deep learning has only come into play in the last few years:

- *Flood of available data*: Nowadays the amount of data available is extremely large.
- *Increased computational power*: There have been continuous improvements in hardware manufacturing that have made it possible to produce devices with a more powerful computational ability. In fact, an extremely important role for processing in the field of machine learning is played by GPUs.
- *Research Development*: There are a lot of new machine learning algorithms and theory developed by researchers.
- *Industry Involvement*: There is much more interest and support provided by companies, such as OpenAI, DeepMind and many others.

2 Data, Features, Models

This section will mainly focus on data, features, and models. As has been mentioned in the previous section the first key element of machine learning is *data*. Then, by processing these data, the second main key will be extracted, the so-called *features*, which are the most relevant part of the data that will be used for learning. Finally, based on the features extracted previously, it will be possible to build the so-called *models* that will represent the summary of the knowledge acquired through the learning process and will be used to perform some actions.

2.1 Learning Process

Based on the results provided in the previous section, it is time to introduce a more complete description of the machine learning pipeline, as illustrated in the image below:

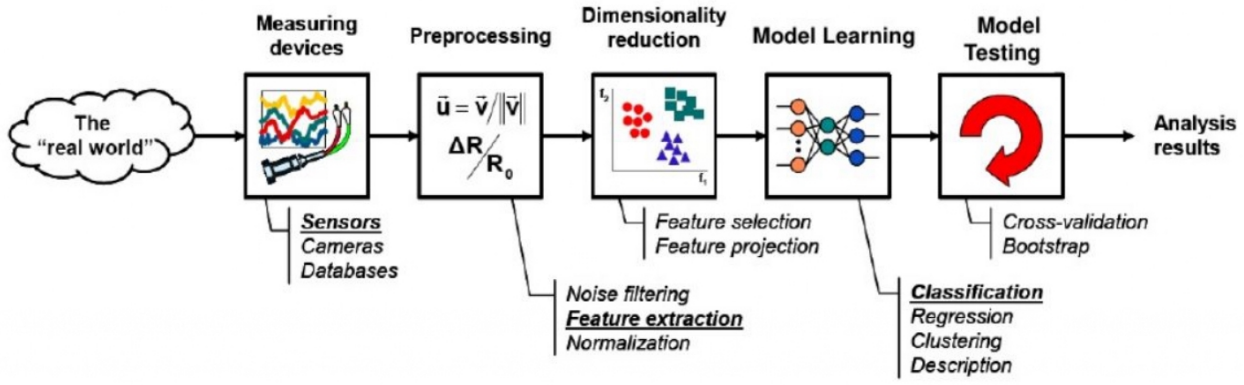


Figure 13: Learning process in detail

So, as can be observed in the image above, the learning process in real-world applications where some kind of machine learning is used is actually quite complex and it is also necessary to understand how to perform each step of the pipeline shown below in order to build an actually working machine learning model:

1. ***Data Source***: This is the actual place where all data are generated. Normally this place is represented by the real world itself.

2. ***Data Collecting***: This is the process by which all data are collected from the data source mentioned in point 1 by using different devices, such as sensors, cameras, and databases.

3. ***Data Preprocessing***: This is a process that depends on the type of the problem which is being addressed. Normally this process consists in some noise filtering, if the data are images or sensor signals, in normalization, if the data are seen as vectors, or in the so-called *feature extraction*, which is achieved by transforming the data into a set of vectors containing relevant information.

4. ***Dimensionality Reduction(Optional)***: Sometimes the features from the previous step are not used directly to produce a model, but are carefully selected to extract the most relevant part through the process called *feature selection*.

5. ***Model Learning***: This is the core step of the machine learning process in which, by applying a particular *learning algorithm*, the actual *model* is generated. The actual type of algorithm that will be applied depends on the particular nature of the problem being solved, such as classification, regression, clustering, description, and many others.

6. ***Model Testing***: Once the model has been generated through a particular learning algorithm, a particular model testing protocol is applied to validate the accuracy of the generated model.

It is also important to mention that another element that adds complexity to the design of machine learning models is the choice of a particular algorithm to use. This is because of the fact that there is a huge number of machine learning algorithms that could be used. Fortunately, there is always a guide that helps in making the right decision, that is the particular nature of the problem to be solved.

At this point, given the general machine learning pipeline described above, it is time to dive into the first most important element of the process which is *data*. The first problem to solve is the fact that the concept of data is quite abstract, but the concrete data that are used in real-world applications can be extremely different form each other. For this reason, it is necessary to define a general procedure that will allow representing data independently of their actual structure. The answer to this problem is called *feature extraction*: that is the process by which each *example* in the *training set* is associated with a data structure, which is normally a simple vector of numbers with cardinality n, that *represents* the relevant information about the example and indicates the *actual form* that is seen by the algorithms. Each such vector takes the name of *feature vector*.

One way of extracting these features is to consider them as *questions that can be asked* about the example, like for instance in the image below with the fruits:

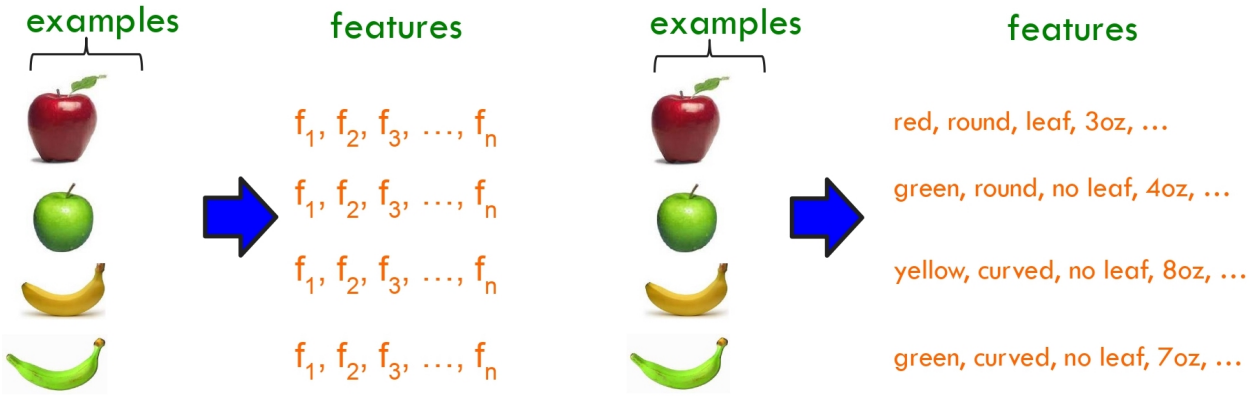


Figure 14: Example of features

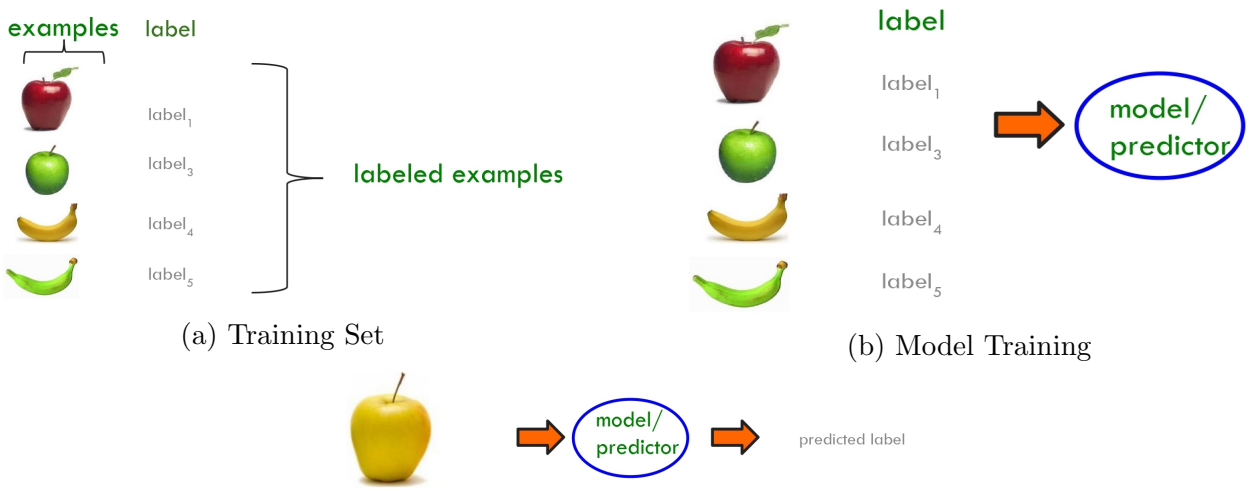
Therefore, the final result of the feature extraction process is the actual *training set*, which is a set of numerical vectors, each of which is the same dimension as the others. Unfortunately, there is a problem with this approach, which is actually the most delicate part of the design of the machine learning pipeline: *how* are these features chosen? There are actually many answers to this question and the best answer should be the one that will ideally produce a set of features that represent as well as possible the original data without any loss of information. Therefore, there is always a risk that, after processing the input data, some information associated with the real data may be lost, which in the case could cause the machine learning algorithm to work improperly.

2.2 Machine Learning Methods

Now it is time to talk about different families of machine learning methods to add a deeper understanding to what has been briefly described in the overview subsection of these notes.

2.2.1 Supervised Learning

The first big family of machine learning methods is the so-called *Supervised Learning*. All methods associated with this family have access to the data *and* some annotations that are associated with these data. These annotations are also called *labels*. Therefore, the training set, that is the input to the learning algorithm, in these cases consists of a set of pairs, each of which consists of an *example*(in the form of a *feature vector*) and its relative *label*. For this reason, all the pairs in the training set are called *labeled examples*. Then this set of labeled examples is processed by the learning algorithm to build a model, also called a predictor, that will be used on new data, which have not been seen during the training phase, to produce labels associated with these data. It is necessary to note that all labels generated by the model will belong to the set of the labels that have been learned from the training set. All this process can be observed in the images below:



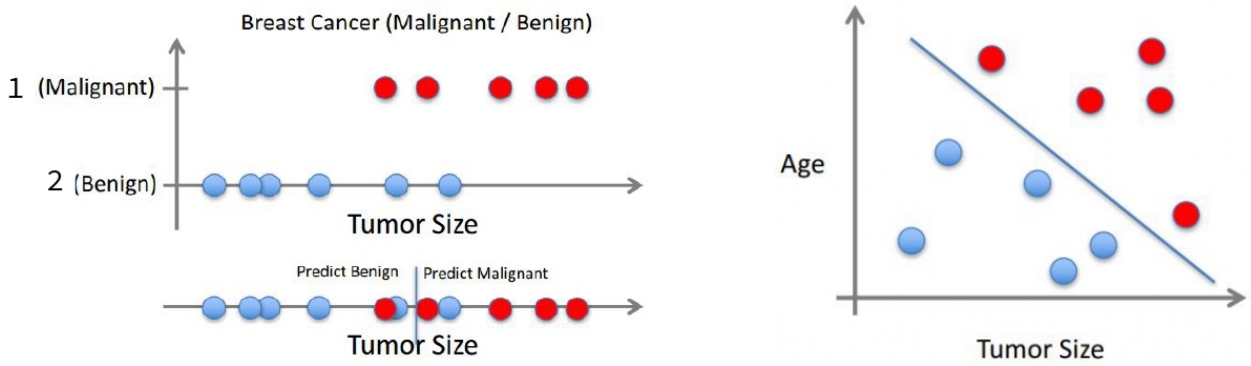
(a) Training Set

(b) Model Training

(c) Model Testing

At this point, it is possible to introduce some major machine learning tasks that use the supervised learning structure defined above:

- ***Classification***: Given a training set $\mathcal{T} = \{(x_1, y_1), ..., (x_m, y_m)\}$ where m is the total number of pairs of labeled examples, the task is to learn a function f, defined as $f : \mathbb{R}^d \to \{1, 2, ..., k\}$ where d is the dimension of the input space and k is the finite number of labels, to predict the label y given the input x. So the dimension of the x component(feature vector) in the pairs of labeled examples and as input to the function f is determined by the value of d, as can be observed in the examples below:



(a) x is one-dimensional($d = 1$)

(b) x is multidimensional($d = 2$)

This classification framework of supervised learning is of course extremely general and can be instantiated for several problems, such as Face Recognition, Character Recognition, Spam Detection, Medical Diagnosis, Biometrics and many others.

- ***Regression***: Given a training set $\mathcal{T} = \{(x_1, y_1), ..., (x_m, y_m)\}$ where m is the total number of pairs of labeled examples, the task is to learn a function f, defined as $f : \mathbb{R}^d \to \mathbb{R}$ where d is the dimension of the input space, to predict the label y given the input x. As with Classification, the dimension of the x component(feature vector) in the pairs of labeled examples and as input to the function f is determined by the value of d. However, the big difference between Classification and Regression consists in the fact that all labels produced by the computed regression function are real-valued, so the set of all possible labels is not finite by definition(i.e. it is continuous). An example of regression is the following:

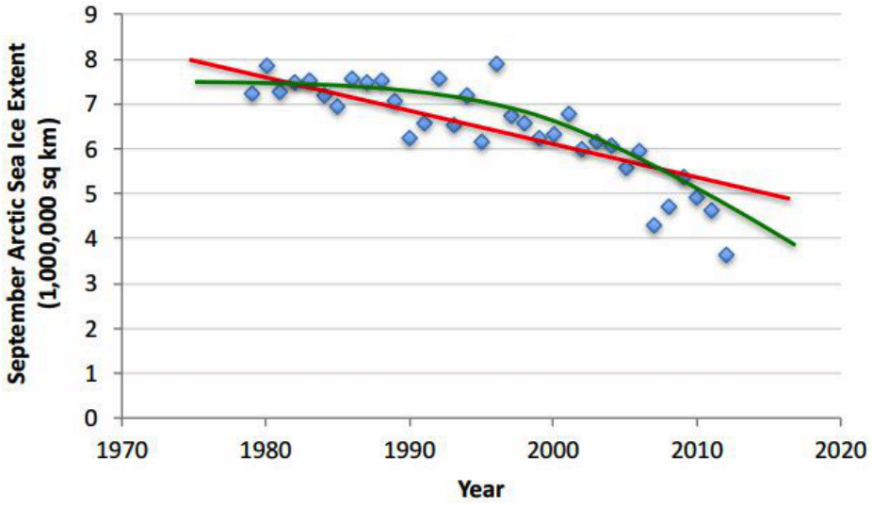


Figure 17: Example of Regression

As with Classification, this regression framework of supervised learning is of course extremely general and can be instantiated for several problems, such as Stock Value Prediction in Finance, Epidemiology, Car/Plane Navigation, Weather Forecasting in the area of Temporal Trends and many others.

- ***Ranking***: This is another supervised learning task in which all the labels in the training set and the ones generated by the computed ranking model are *ranks*, which are values indicating the sorting position/relevance of the elements. An example of ranking is search query evaluation in web browsers where, given a query and a set of web pages, a ranking is computed according to the ranks of the web pages, with the most relevant web sites appearing at the top. As with the previous two tasks, this ranking framework is extremely general and can be instantiated for several problems, such as User Preference(e.g. Netflix "My List" for movie queue ranking), Image Retrieval, Flight Search, Search in general and many others.

2.2.2 Unsupervised Learning

The second big family of machine learning methods is the so-called *Unsupervised Learning*. As with supervised learning, all methods associated with this family have access to the data *but* there are *no* annotations associated with these data. Therefore, the training set in this case consists only of a set of *examples*(always in the form of *feature vectors*) without any associated labels. Then this set of examples is processed by the learning algorithm to build a model consisting of a *hidden structure* behind the input data.

At this point, it is possible to introduce some major machine learning tasks that use the unsupervised learning structure:

- ***Clustering***: Given a training set $\mathcal{T} = \{x_1, ..., x_m\}$ where m is the total number of examples, the task is to output a hidden structure behind the input data. In this particular case the discovered structure is represented by the so-called *clusters*, which are perfectly illustrated in the following image:

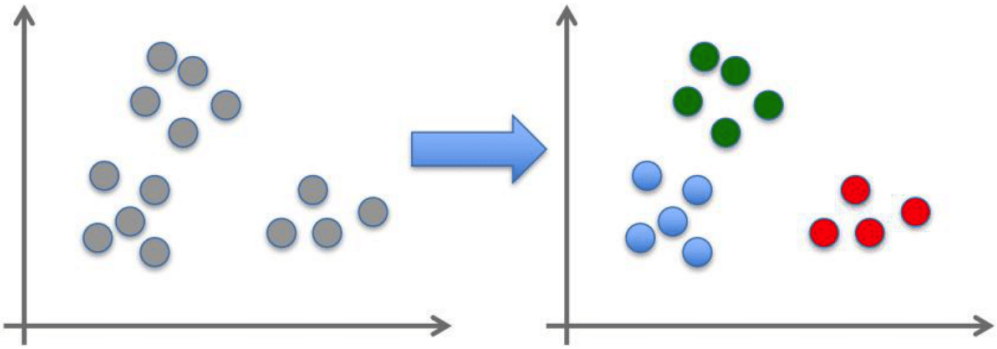


Figure 18: Example of Clustering

  This clustering framework of unsupervised learning is of course extremely general and can be instantiated for several problems, such as Social Network Analysis, Genomics, Image Segmentation and many others.

- ***Anomaly Detection***: This is another unsupervised learning task in which the training set consists of events or objects and the task consists in detecting those events or objects that are unusual or atypical.

  This anomaly detection framework of unsupervised learning is of course extremely general and can be instantiated for several problems, such as Credit Card Fraud Detection, Video Surveillance and many others.

- ***Dimensionality Reduction***: This is another unsupervised learning task in which the training set consists of data characterized by a certain number of dimensions (i.e. variables) and the task is to reduce these dimensions in order to get a set of data characterized by a reduced number of variables while preserving the most important relationships between the data as they were in the initial set. This particular task can be observed in the following image:

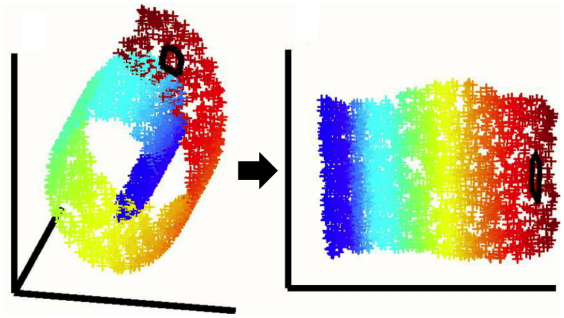


Figure 19: Example of Dimensionality Reduction

This dimensionality reduction framework of unsupervised learning is of course extremely general and can be instantiated for several problems, such as Output Inspection in the case of a classification algorithm and many others.

2.2.3 Reinforcement Learning

The third big family of machine learning methods is the so-called *Reinforcement Learning*. The big difference between this family and the previous two consists in the fact that there is no training set in reinforcement learning but there is only a numerical performance score as its guidance. Therefore, in this case there is a totally different learning scheme, which is based on the following idea: an *agent* learns from an *environment* by interacting with it through some *actions* that make it receive *rewards* and make it modify its current *state*. This idea is clearly illustrated in the following image:

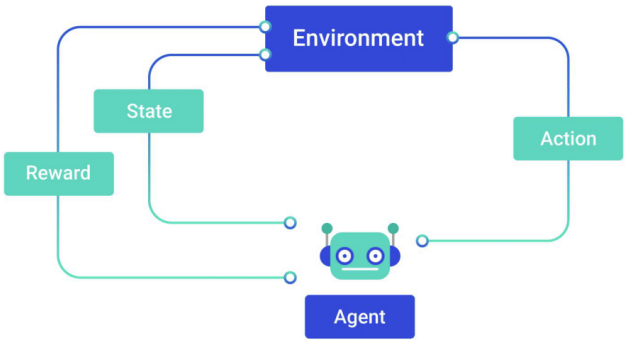


Figure 20: Reinforcement Learning Scheme

It immediately becomes clear that this learning scheme is highly *interactive* because the agent, in order to learn some useful information, continuously completes *sequences of states/examples* and gets *rewards* after completing those sequences. In this way, based on the values of the rewards, the agent learns to predict the *action* to take for each individual state/example in order to achieve its final goal, which is an optimal, or nearly-optimal, way to maximize the amount of gained rewards. An example of a sequence evaluation is illustrated in the following image:

left, right, straight, left, left, left, straight	GOOD
left, straight, straight, left, right, straight, straight	BAD
left, right, straight, left, left, left, straight	18.5
left, straight, straight, left, right, straight, straight	-3

Figure 21: Example of a sequence evaluation

Some famous examples of the application of reinforcement learning are for instance the Backgammon and Go board games and the computer game Atari Breakout, in which you try to learn winning sequences.

2.2.4 Other Learning Variations

Besides the three largest families of machine learning methods, there are other families that will not be covered in these notes. However, it is always useful to mention *some* of them because of their usage in different areas:

- ***Semi-supervised Learning***: This is a family of hybrid machine learning methods that involve small datasets of labeled examples(supervised learning) and large datasets of examples that don't contain any labels(unsupervised learning).

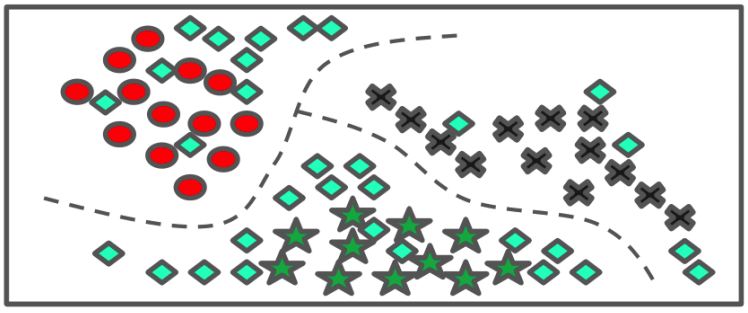


Figure 22: Example of Semi-supervised Learning

- ***Active Learning***: This is a family of machine learning methods that are characterized by an incremental learning scheme and by some human supervision(some useful feedback) which is involved only in the case of the most difficult examples.

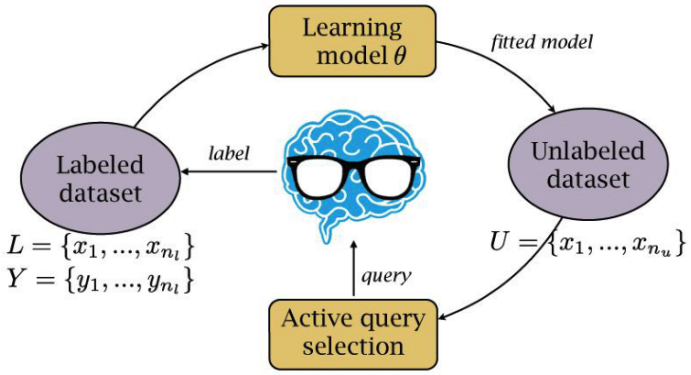


Figure 23: Example of Active Learning

Some learning variations also derive from *how* the data are retrieved:

- ***Online Learning***: There is a continuous process during which the incoming data are immediately used to update the model.
- ***Offline Learning***: The data, such as training sets, are available offline and are used to train a model that will then be used on new unknown data.

Other learning variations also derive from the *type* of a model:

- ***Generative vs Discriminative Learning***: These two different types will be covered in the particular case of supervised learning later in these notes.
- ***Parametric vs Non-parametric Learning***: Some models may or may not contain some parameters that are learned during the training phase.

2.3 Intro to Data Generating Distributions

Now it is time to talk more deeply about one of the most relevant and critical elements of machine learning, that is *features* and their *generation*. As mentioned previously, the training set is made of examples(labeled in the case of supervised learning) that are represented in the form of features and then the model, that is generated during the training phase, makes predictions **based on these features**. The testing set is also made of examples that are represented in the form of features, but in this case, no answers are provided, since the answers will be generated by the model, which will process the features of each example to produce the predictions(labels in the case of supervised learning).

The natural question that arises from this summary is how the training and testing sets should be generated. The natural answer to this question is that **the same feature extraction algorithm** should be applied to generate both the training and testing sets, but this is not enough. The fact is that the result of a prediction will strongly depend on the features that have been seen, so the examples in the testing set should contain features that are *similar* to some features in some examples in the training set. This is an intuition that has to be formalized and satisfied in order to be able to **generalize from the training set**. So the implicit assumption here is that the examples in the testing set are *in some way similar* to the examples in the training set. This is not always the case, but sometimes it will be necessary to assume that it is.

At this point, it is necessary to formalize the intuition described above and this can be achieved by modeling the similarity between the training and testing sets through a *probabilistic model* of learning. This probabilistic modeling implies **the most important assumption of machine learning**, that is the fact that **it is possible to learn** because it is **assumed** that **both** the training **and** testing sets are generated based on the **same** underlying probability distribution, which is **unknown** and called in this context **Data Generating Distribution(DGD)**. So, thanks to this assumption, it is possible to build systems that can generalize between the training and testing sets by starting to consider the different features of the examples in the training and testing sets in terms of probability, as can be observed in the following image:

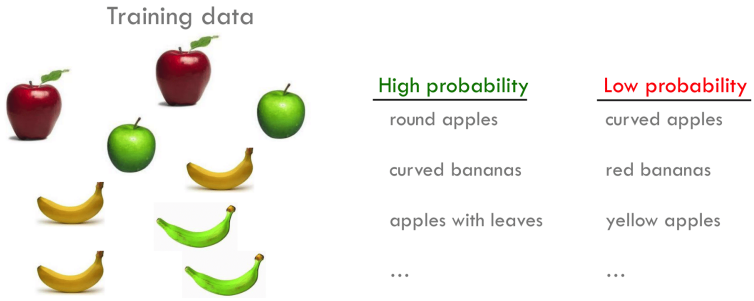


Figure 24: Example of probability values

In order to start getting familiar with the concept of data generating distribution it could be useful to observe the following image, which represents a data generating distribution with different *likelihood values of sampling*, for every fruit category, in terms of different amounts of fruits:

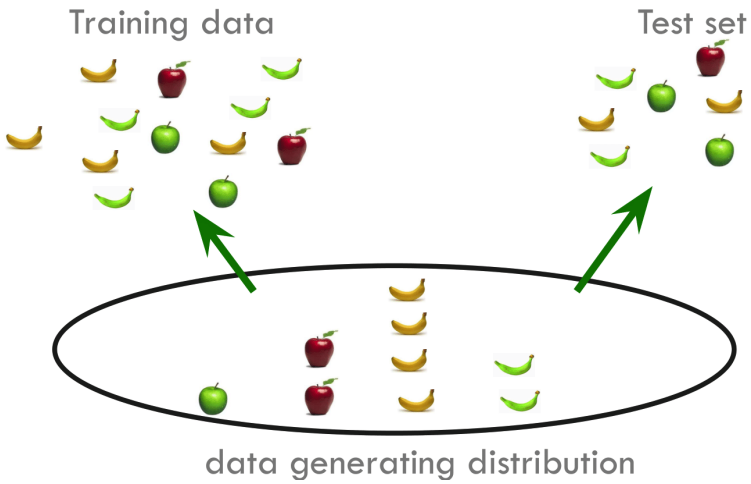
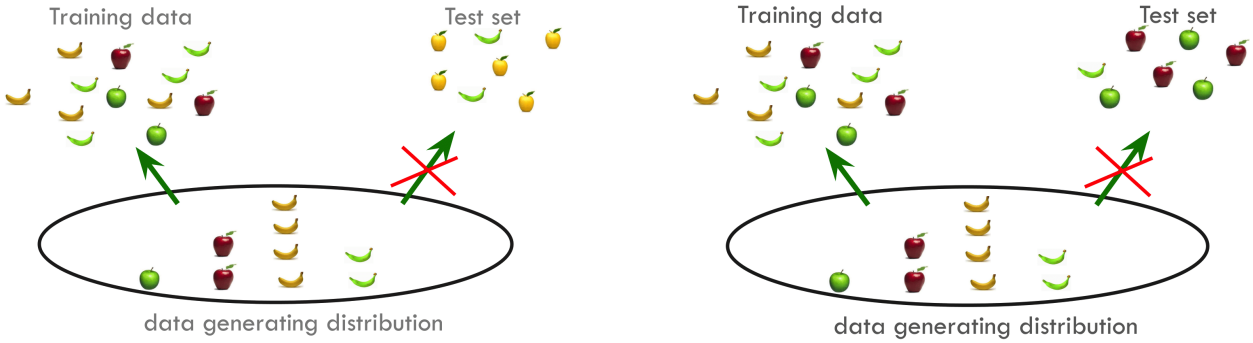


Figure 25: Example of a DGD

It is also extremely useful to observe some counterexamples that clarify even more the importance of the fact that both the training and testing sets must be generated based on the same probability distribution:



(a) Testing Set with new fruits

(b) Testing Set without bananas

Figure 26: Training and Testing Sets generated by different DGDs

2.4 Applied Machine Learning

In this subsection, the refined version of the machine learning process will be presented, with a particular attention on its usage in real-world applications. So, the following are the five main ingredients to build a working machine learning system:

- ***Task***: First of all, it is necessary to formalize the task that is being solved into one of the common tasks of machine learning, for example when the task of recognizing a bird in a photo becomes a classification task, which is well-known in machine learning.

- ***Data***: It is always crucial to have a proper set of data in order to correctly solve a given task. The importance of data in machine learning will always be a fundamental aspect of the whole process.

- ***Model and Hypothesis Space***: In general, it is always possible to choose, from a large family of machine learning models, a particular model structure that will be suitable for the task.

- ***Objective Function***: It is always necessary to define an objective function that is directly related to the performance measurement of the chosen model.

- ***Learning Algorithm***: Based on the choice of a particular model and performance measurements given by the objective function, it is possible to apply some machine learning algorithm in order to produce the desired model.

At this point, it is useful to dive into each of the five elements mentioned above in order to understand more clearly the importance of their roles.

2.4.1 Task

In general a task can be identified with a *set of functions* $\mathcal{F}_{task} \subset \mathcal{Y}^{\mathcal{X}}$, where $\mathcal{X}$ is the input space and $\mathcal{Y}$ is the output space, that can potentially solve a problem. So it consists of functions, in the form of $f : \mathcal{X} \to \mathcal{Y}$, which assign an input $x \in \mathcal{X}$ an output $y \in \mathcal{Y}$. Obviously, the nature of $\mathcal{X}, \mathcal{Y}$ and $\mathcal{F}_{task}$ will depend on the type of task to be solved, as can be observed from the tasks formalized below:

- ***Classification***: The output space $\mathcal{Y}$ is discrete, so each input $x \in \mathcal{X}$ is mapped through the function f onto $f(x) \in \mathcal{Y} = \{c_1, ..., c_k\}$.

- ***Regression***: The output space $\mathcal{Y}$ is continuous, so each input $x \in \mathcal{X}$ is mapped through the function f onto $f(x) \in \mathcal{Y} \subseteq \mathbb{R}$.

- ***Density Estimation***: An unsupervised learning task that consists in calculating the underlying probability distribution $f \in \Delta(\mathcal{X})$ over the data in $\mathcal{X}$.

- ***Clustering***: An unsupervised learning task that consists in calculating a function $f \in \mathbb{N}^{\mathcal{X}}$ that assigns each input $x \in \mathcal{X}$ a cluster index $f(x) \in \mathbb{N}$, so all points that are mapped to the same index form a cluster.

- ***Dimensionality Reduction***: An unsupervised learning task that consists in calculating a function $f \in \mathcal{Y}^{\mathcal{X}}$ that maps each higher dimensional input $x \in \mathcal{X}$ to a lower dimensional embedding $f(x) \in \mathcal{Y}$, where $dim(\mathcal{Y}) \ll dim(\mathcal{X})$.

2.4.2 Data

Different types of tasks can be performed on different types of data, which can vary from movie records taken from IMDb or environmental changes registered and collected by electronic devices to a simple collection of fruits. The important aspect here is the fact that

all these significantly different types of data are represented in the same way, that is, in terms of their most important characteristics, which are called *features*. Therefore, these features, which are normally grouped into vectors, represent how the input data are actually visualized by machine learning algorithms. Unfortunately, as explained earlier, this feature extraction process almost always results in a loss of information about the original form of the data.

Besides the formal way of representing data in terms of vectors of features, the other extremely important concept about data is the so-called *data generating distribution*. This data generating distribution is generally *unknown* and represents a distribution of probabilities by which the data are generated. From this point of reading onwards, the data generating distribution will be called p_{data} and will be formally defined as follows:

- ***Classification and Regression***:

$$p_{data} \in \Delta(\mathcal{X} \times \mathcal{Y})$$

- ***Density Estimation, Clustering and Dimensionality Reduction***:

$$p_{data} \in \Delta(\mathcal{X})$$

Since this data generating distribution is unknown, the only way to access it is through *sampling*. So, in practice it is assumed that training data, validation data and test data are generated through sampling from the same underlying probability distribution and only in a similar case the final model will probably work well. A typical machine learning pipeline in which the assumption about the data generating distribution is satisfied can be observed in the image below:

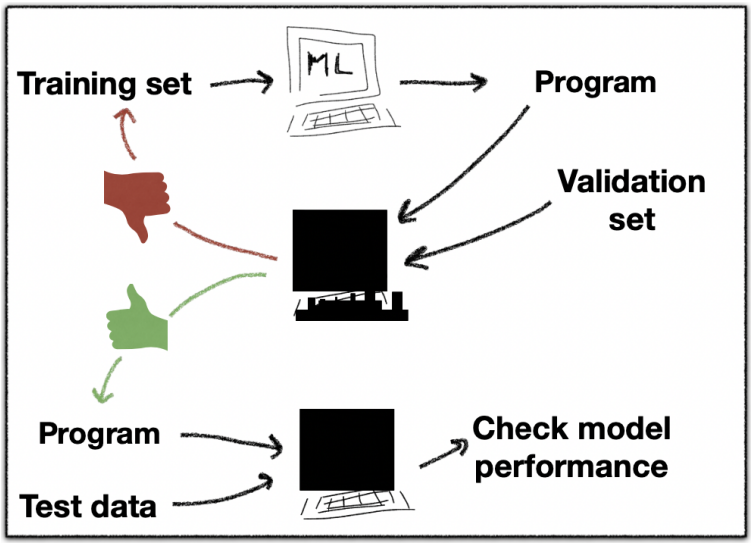


Figure 27: Practical Machine Learning Pipeline

In machine learning, besides the importance of the feature extraction process and the assumption about the data generating distribution of the datasets, a particular role is also played by the so-called **Training Set Design**. This aspect is particularly important because the failure of a machine learning algorithm is often caused by a bad selection of training samples. The issue is that sometimes some unwanted correlations are introduced into the dataset, from which the algorithm could derive wrong conclusions. This problem is perfectly described by the following probably invented story:

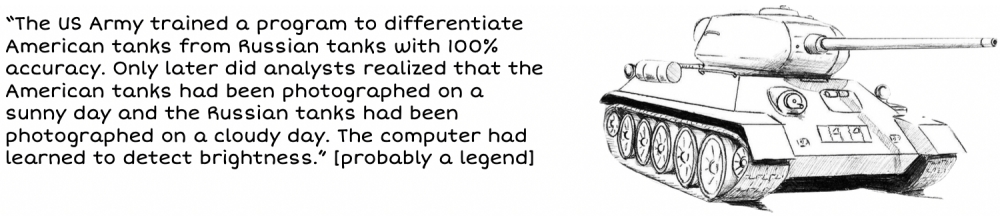


Figure 28: Why the training set design is so important

2.4.3 Model and Hypothesis Space

At this point, once the task has been identified and the datasets have been properly built, it is necessary to talk about the next two key concepts that are called **Model** and **Hypothesis Space**. The model is basically a *program* that solves the task, so it is the *implementation* of the above-mentioned function $f \in \mathcal{F}_{task}$ that can be tractably computed. The hypothesis space is a *set of models* $\mathcal{H} \subset \mathcal{F}_{task}$ where a particular learning algorithm seeks a solution, that is, a model. So, in general, the set of models from which to choose is a subset of all possible models that could solve a given task, since the number of all possible algorithms and consequently of all possible models produced by these algorithms is particularly big and it should be clear that some algorithms are yet to be discovered. The following image perfectly describes what has just been explained:

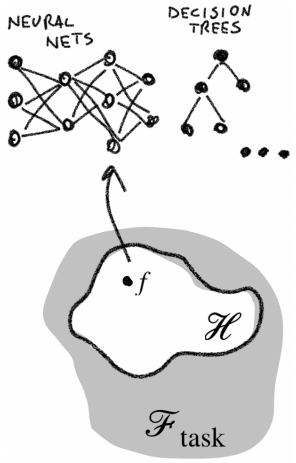


Figure 29: Model and Hypothesis Space

From the formal and practical point of view, for example in the case of *Polynomial Curve Fitting*, the general form of a model that solves a given task is the following:

$$f_w(x) = \sum_{j=0}^{M} w_j x^j$$

So, the learning process, in this case, consists in discovering the *best function* f(at this point of reading it is not clear what this means), which actually means that the learning process consists in discovering the *best parameters* $\{w_0, ..., w_M\}$ of that function.

The hypothesis space, on the other hand, is formally defined as follows:

$$\mathcal{H}_M = \{f_w : w \in \mathbb{R}^M\}$$

This means that the hypothesis space is the set of all those functions, parameterized on values $w_0, ..., w_M$, which, for some *fixed* polynomial degree $M \in \mathbb{N}$, solve the given task. An example of different polynomial models with different degrees can be observed in the image below:

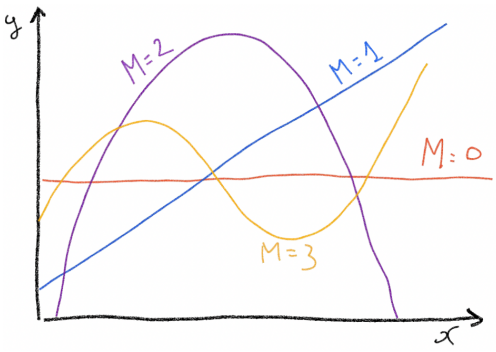


Figure 30: Polynomial Models

2.4.4 Objective Function

At this point, it is necessary to mathematically formalize the abstract property of a model of being the best choice to make among all possible models and this is done by introducing the concept of **Generalization Error Function** $E(f; p_{data})$ that determines how well a solution $f \in \mathcal{F}_{task}$ fits some given data. So, this particular objective function basically guides the selection of *the best ideal solution* in $\mathcal{F}_{task}$. This best ideal solution is formally defined as follows:

$$f^* \in \arg \min_{f \in \mathcal{F}_{task}} E(f; p_{data})$$

An example of the best ideal solution to a given task can be observed in the following image:

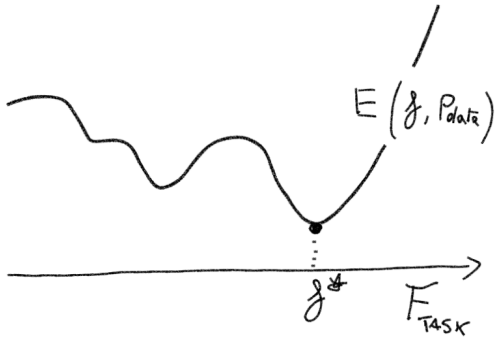


Figure 31: Best Ideal Model in $\mathcal{F}_{task}$

Unfortunately, the biggest problem with this approach is the fact that **the search space $\mathcal{F}_{task}$ is too large and the data generating distribution p_{data} is unknown**, so there is no available procedure that would allow calculating the best ideal model. This is the reason why, at first, it is necessary to restrict the search space $\mathcal{F}_{task}$ to the hypothesis space $\mathcal{H} \subset \mathcal{F}_{task}$ and then to calculate the best solution that can be implemented and evaluated in a tractable way within this restricted space, always be considering the same objective function $E(f; p_{data})$. In this way, *the best feasible solution* would be formally defined as follows:

$$f^*_{\mathcal{H}} \in \arg\min_{f \in \mathcal{H}} E(f; p_{data})$$

An example of the best feasible solution found within the hypothesis space can be observed in the following image:

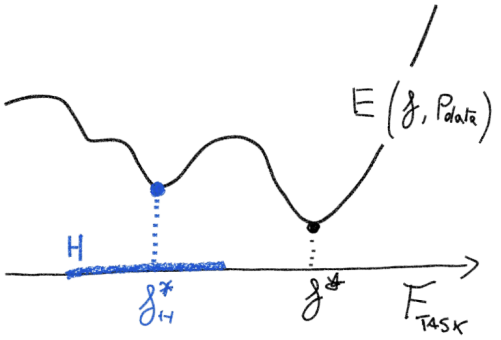


Figure 32: Best Feasible Model in $\mathcal{H}$

Unfortunately, even in this case it is impossible to calculate the best solution, since **the data generating distribution p_{data} is still unknown**. This is the reason why it is necessary to work only on a set of observations that are *sampled* from the data generating distribution, that is, a **Training Set** $\mathcal{D}_n = \{z_1, ..., z_n\}$ with $z_i \sim p_{data}$, where in the case of polynomial curve fitting $z_i = (x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$. In this way, *the best actual solution* would be formally defined as follows:

$$f^*_{\mathcal{H}}(\mathcal{D}_n) \in \arg\min_{f \in \mathcal{H}} E(f; \mathcal{D}_n)$$

Moreover, in this case it is no longer possible to consider the same objective function, but it is necessary to introduce a new objective function $E(f; \mathcal{D}_n)$, which is called **Training Error Function**. The important aspect here is that the two error functions should be as similar as possible.

An example of the best actual solution found within the hypothesis space can be observed in the following image:

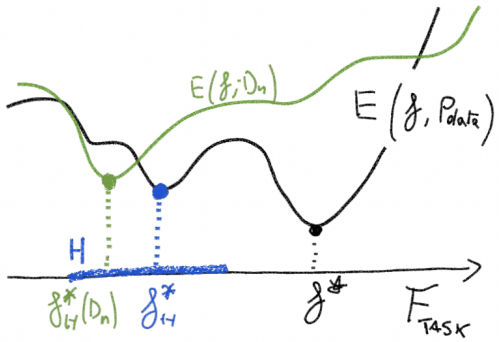


Figure 33: Best Actual Model in $\mathcal{H}$

At this point, it is possible to dive into the actual mathematical form that the generalization and training error functions have. Typically the generalization and training error functions can be written in terms of a **pointwise loss function** $l(f; z)$ measuring the error incurred by f on the training example z. So, the two functions would have following forms:

$$E(f; p_{data}) = \mathbb{E}_{z \sim p_{data}}[l(f; z)]$$

and

$$E(f; \mathcal{D}_n) = \frac{1}{n} \sum_{i=1}^{n} l(f; z_i)$$

This pointwise loss function is basically computed by comparing the prediction $f(x)$ of the model, associated with the input x, and the so-called **ground truth** z, also associated with the input x. It is also important to notice that there are several possible implementations of this pointwise loss function, which will be analyzed later in these notes. For example, in the case of polynomial curve fitting, the implementation of the pointwise loss function $l(f; z_i) = l(f; (x_i, y_i))$ is $[f(x_i) - y_i]^2$, so the training error function becomes as follows:

$$E(f; \mathcal{D}_n) = \frac{1}{n} \sum_{i=1}^{n} [f(x_i) - y_i]^2$$

and takes the name of **Mean Squared Error**. The meaning of this particular implementation can be observed in the following image:

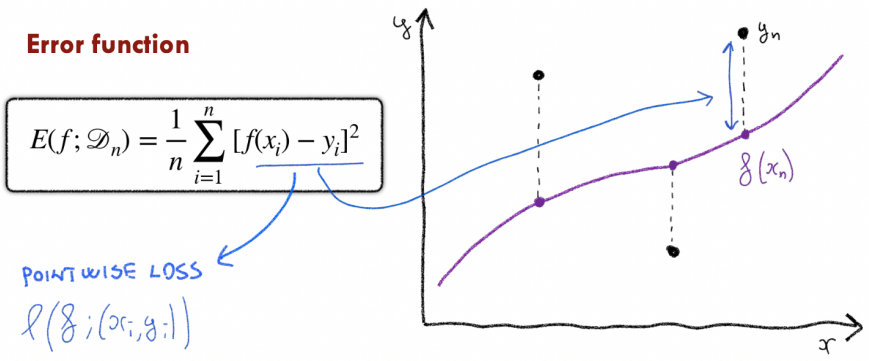


Figure 34: Mean Squared Error

So, in the case of polynomial curve fitting, the best actual model would be formalized as follows:

$$f^*_{\mathcal{H}_M}(\mathcal{D}_n) \in \arg\min_{f\in\mathcal{H}_M} E(f;\mathcal{D}_n)$$

which is actually equivalent to f_{w^*} where w^* is:

$$w^* \in \arg\min_{w\in\mathbb{R}^M} \frac{1}{n}\sum_{i=1}^{n}[f_w(x_i) - y_i]^2$$

At this point, it is necessary to emphasize that the last equation mentioned above *can be actually computed* and in general only requires solving a linear system of equations.

However, there is still one issue that has to be addressed. This issue consists in the fact that, due to optimization difficulties encountered during the execution of the learning algorithm, it might happen that the computed solution is only a *local minimum* $\hat{f}^*_{\mathcal{H}_M}(\mathcal{D}_n)$ and not a *global minimum* $f^*_{\mathcal{H}_M}(\mathcal{D}_n)$ that has been described so far. This unfortunate issue can be observed in the following image:

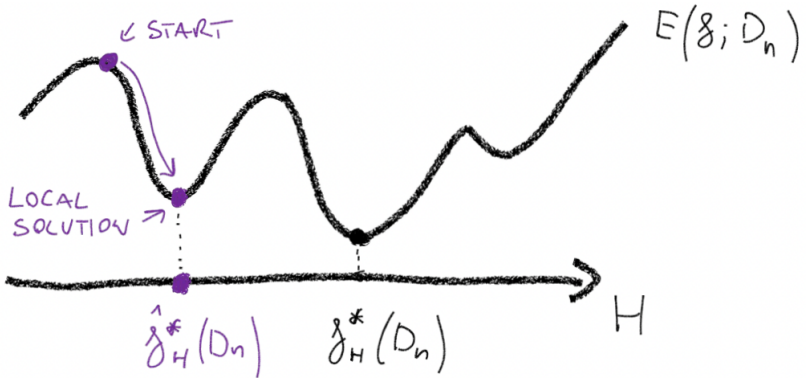


Figure 35: Local and Global Minima

So, at this point, it could be useful to summarize the differences between all the different models that have been discussed so far by illustrating them in the following image:

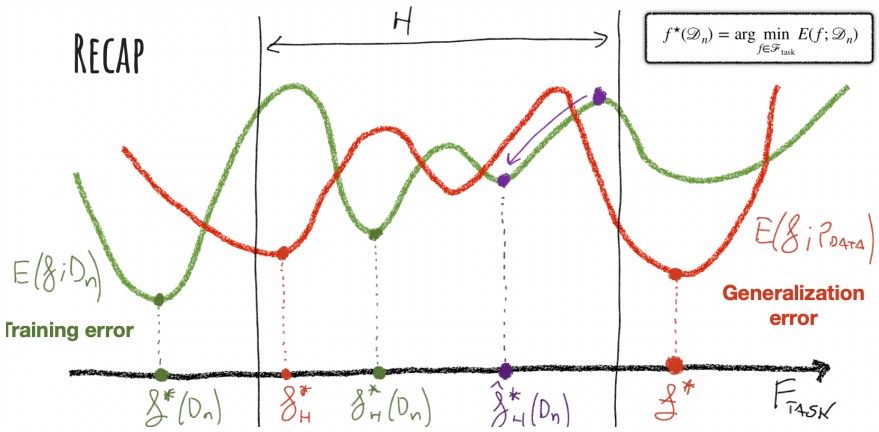


Figure 36: Summary of all types of models

At this point of this already long discussion about models, it is necessary to talk about two other key concepts of machine learning that are **Underfitting** and **Overfitting**:

- ***Underfitting***: The situation of underfitting is observed when the value of the training error function is *large*. This means that there is a big discrepancy between the accuracy of the model *that has been computed* and the accuracy of the best model that *can be computed* and that minimizes the training error. In other words, it is impossible to learn well enough from that particular training set. The concept of underfitting can be summarized by the following image:

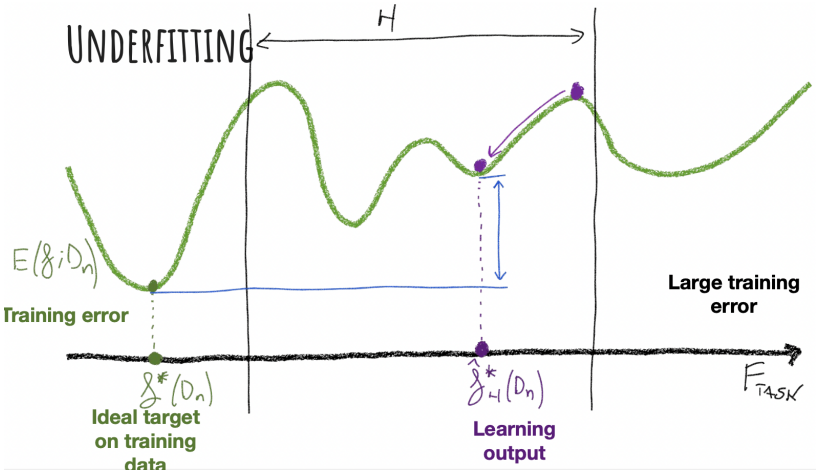


Figure 37: Example of Underfitting

- ***Overfitting***: The situation of overfitting is observed when the computed model has learned too well from a particular training set, so it performs exceptionally well on that dataset, but for this very reason there is a *large generalization gap*, which means that the model is not able to properly generalize to new unknown data, which are different from those seen in the training set. The concept of overfitting can be summarized by the following image:

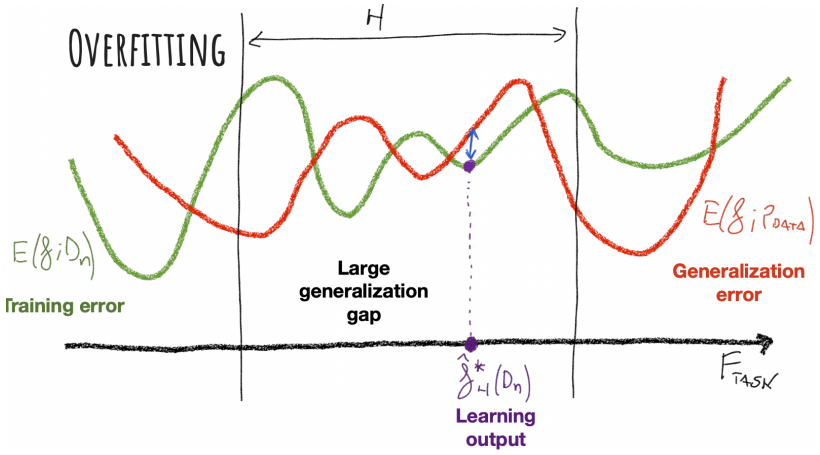


Figure 38: Example of Overfitting

Now is the time to talk about other types of errors that are always present and that depend on the choice of a particular training set, a particular hypothesis space and the inherent variability of the predictions made on new unknown data:

- ***Estimation Error***: This is the error induced by learning on a *particular data sample*. Therefore, it indicates the difference between the computed model and the best model in the hypothesis space that minimizes the generalization error. This difference can be observed in the following image:

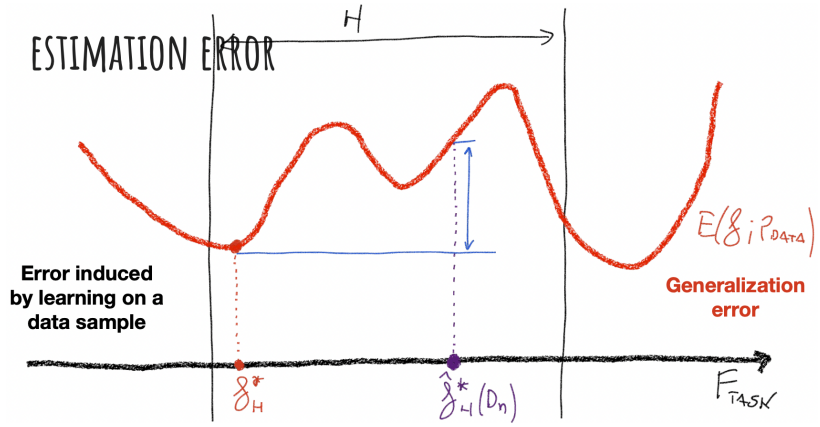


Figure 39: Example of Estimation Error

- ***Approximation Error***: This is the error induced by the choice of *a particular hypothesis space* $\mathcal{H}$. Therefore, it indicates the difference between the best model in the chosen hypothesis space that minimizes the generalization error and the best ideal model that minimizes the generalization error. This difference can be observed in the following image:

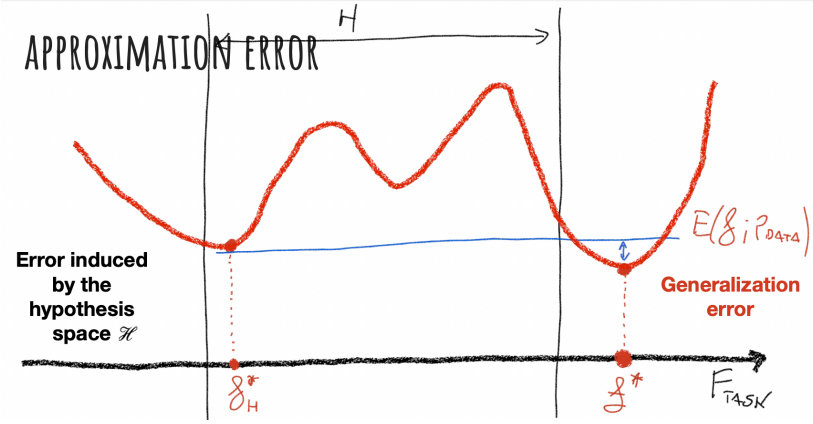


Figure 40: Example of Approximation Error

- ***Irreducible Error***: This is the error due to the *inherent data variability*. Therefore, it basically means that there will *always* be some error between the predictions and the ground truths. This type of error can be observed in the following image:

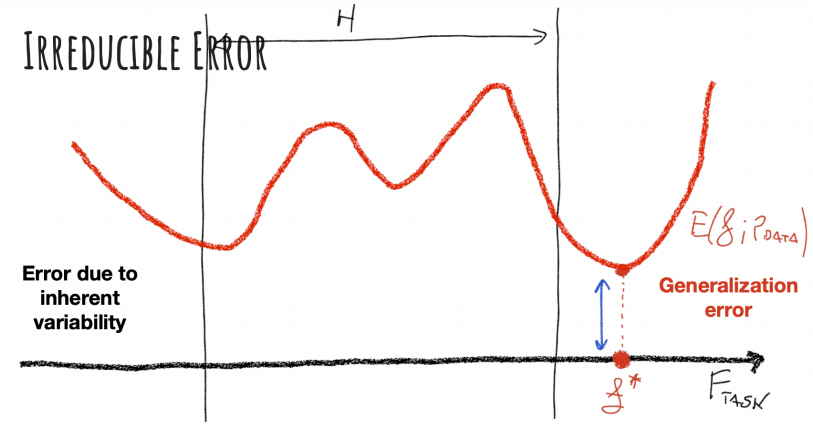


Figure 41: Example of Irreducible Error

At this point of reading, it is clear that the generalization error can not be computed because the data generating distribution p_{data} is unknown. Fortunately, it is possible to approximately bypass this problem by using some data sampled from that underlying distribution, so that, during the execution of the learning algorithm, the training phase allows calculating the best model and the validation phase allows calculating the best *hyperparameters* for that model. Once this sequence of operations has been performed, it is **assumed** that the training error calculated on the final model should provide an acceptable estimation of the generalization error. For this reason it is believed that this final model will perform quite well on the test set. Therefore, it is possible, for example in the case of polynomial curve fitting, to select the best model by considering the estimation of the generalization error and by taking into account the concepts of underfitting and overfitting, as can be observed in the following image:

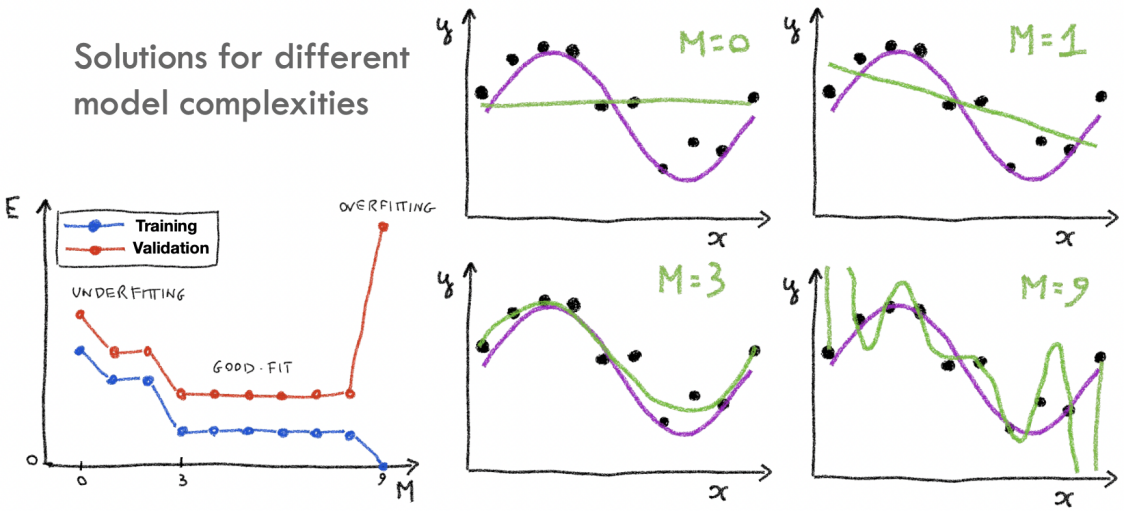


Figure 42: Example of Model Selection

However, it is becoming more and more interesting to understand how to improve the generalization aspect of the model, and this can be done by considering the following techniques, which will be discussed in more detail later in these notes:

- ***Minimal Training Error Avoidance***: In general, it is recommended trying to avoid the minimal training error in order to avoid overfitting.
- ***Model Capacity Reduction***: In general, it is recommended reducing the capacity of the model in order to avoid overfitting, as can be observed in the image above in the case of a polynomial of degree 9.
- ***Regularization***: In this case, the objective function is integrated with a regularization term that allows penalizing solutions that are too complex and that tend to suffer from overfitting.
- ***Noise Injection***: By injecting some data noise into the learning algorithm, the model starts learning new features that will be useful for the future predictions.
- ***Convergence Avoidance***: In some cases, but especially in the case of a learning algorithm used to generate artificial neural networks, it is useful to stop the learning algorithm before it reaches the *convergence*, which means that the final model will not learn too much from the training set and will probably be able to generalize more to the test set.

At this point, it is worth diving into the regularization approach mentioned above to understand more in detail what it means to integrate the objective function with a regularization term. It basically means to add to the objective function $E(f; \mathcal{D}_n)$ a term $\lambda_n \Omega(f)$, where λ_n is a **hyperparameter** and is called **tradeoff parameter** because it defines a tradeoff between the original objective function and the new term in the sense that, by trading off both the objective function and the addictive term, one chooses to be more addictive to the training data or to enforce generalization to prevent overfitting. Therefore, the *new* objective function becomes as follows:

$$E_{reg}(f; \mathcal{D}_n) = E(f; \mathcal{D}_n) + \lambda_n \Omega(f)$$

and this concept can be observed even more clearly in the following image:

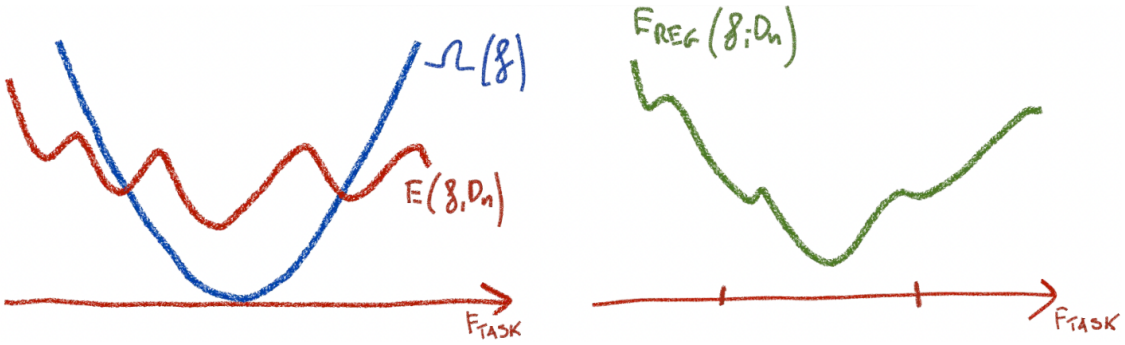


Figure 43: Concept of Regularization

For example, in the case of polynomial curve fitting, one type of regularization is called **L2 Norm Regularization**, where the sum of the squares of the polynomial coefficients is minimized along with the original objective function, so that polynomials with large coefficients are penalized. Therefore the training error function becomes as follows:

$$E_{reg}(f_w; \mathcal{D}_n) = \frac{1}{n}\sum_{i=1}^{n}[f_w(x_i) - y_i]^2 + \frac{\lambda}{n}\|w\|^2, \text{ where } \|w\|^2 = \sum_{j=1}^{M} w_j^2$$

and an example of this regularization can be observed in the following image:

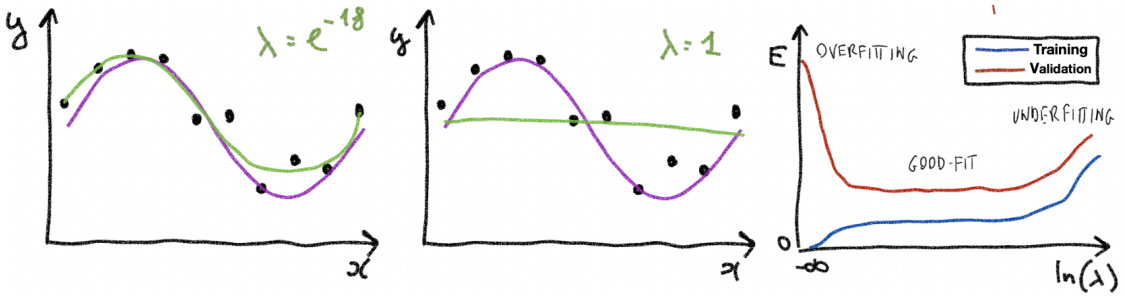


Figure 44: Example of Regularization

where, the parameter λ will be set by considering the performance of the model on the validation set. In this image it is quite clear that the parameter λ somehow *regulates* the tradeoff between overfitting and underfitting, so that, for large values of λ, the model will tend to underfit and, for very small values of λ, the model will tend to overfit. Therefore, the right value for λ will lie somewhere in between those large and very small values.

However, there are also other techniques that can improve the generalization ability of the final model and these techniques are the following:

- ***Increased Training Set Size***: This basically means that the more training data are available the more likely it is that the final model will be able to generalize to new unknown data. Unfortunately, this is not always possible.
- ***Data Augmentation***: This approach suggests *augmenting* the training set by applying some random *transformations*(in the case of images these can be flips and rotations) to the data already available in the training set in order to produce new data that will hopefully make the final model more capable of generalizing.
- ***Ensemble Learning***: This approach suggests combining the predictions of multiple, uncorrelated models during the test phase in order to have a combined prediction that will probably provided a better result.

As the last deepening of this long and complicated subsection, it is worth diving into the increased training set size approach where the size of the training set is increased to improve the generalization capability. This approach is quite interesting since, even if the model flexibility is high(as in the case of polynomial curve fitting where the polynomial degree is high), the training set is also very large, so the overall model is quite accurate. The reason why this happens is that, as the size of the available data increases, the training error tends to become more and more similar to the generalization error, such that the following relation is obtained:

$$E(f; \mathcal{D}_n) \to E(f; p_{data}) \text{ as } n \to \infty$$

This result can be observed even more clearly in the following image:

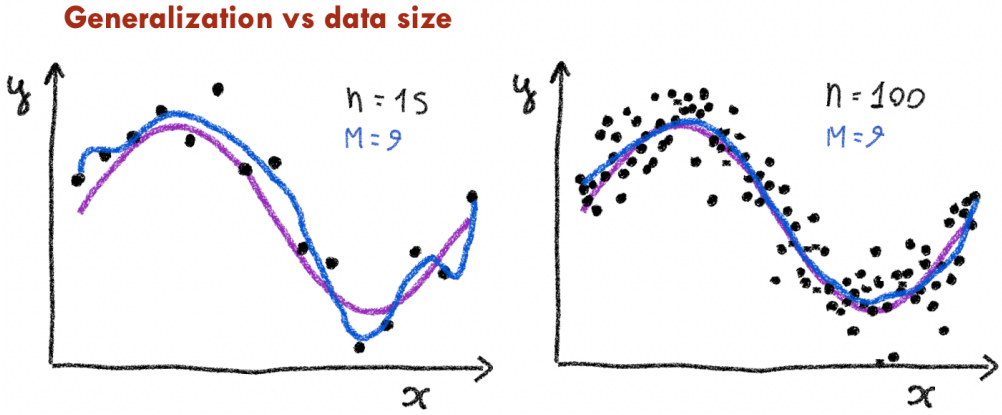


Figure 45: Generalization vs Data Size

At this point, there should be a discussion about learning algorithms, however, since the number of these learning algorithms is considerably large, it is better to postpone this discussion to the next section.