***Is Learning Feasible?***

In this chapter we will explore the relation between probability and machine learning, we will start by giving an example of a probabilistic approach on generalization then relate that example to the learning problem. We saw in the previous chapter how the basic mechanics on machine learning relate to each other, and we also saw that the premise of machine learning is to discover the underlying system (the unknown target function f) using just a finite set of observations. This is a major point, but what does guaranty that we can generalize an infinite real set using a finite training set?

The target function f is the object of learning. The most important assertion about the target function is that it is unknown. We really mean unknown. When we get the training data D, we know the value of f on all the points in D. This doesn't mean that we have learned f, since it doesn't guarantee that we know anything about f outside of D. We know what we have already seen, but that's not learning. That's memorizing. It is important to distinguish between memorizing and learning, in learning at the end of the training phase, when we get the hypothesis g, we disregard the data, why? Because we used it to extract the underlying process that governs its distribution, while in memorizing, the model tends to panic in facing new observations (we want our model to learn not to memorize i.e. the dataset is not important, i.e. if I use a dataset on cancer patients from Michigan to do some classification tasks it should give me the same performance if I used a dataset from cancer patients in jbel-me9re3).

# Example on how Learning is not feasible:

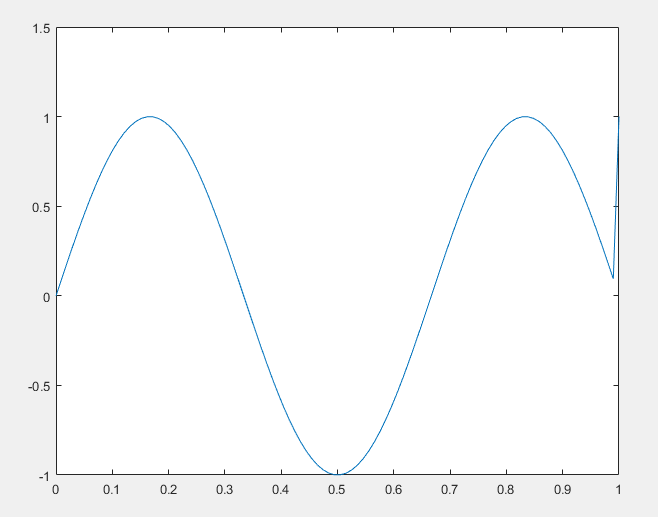
Let’s see we want to approximate some function L but we only know its values on a finite set D

Figure 1 an example of a target function L in D=[0,1]

The goal of learning is to infer L outside the known data D. But there are an infinite amount of functions that takes the same value as L in the known domain but takes different values outside of that domain

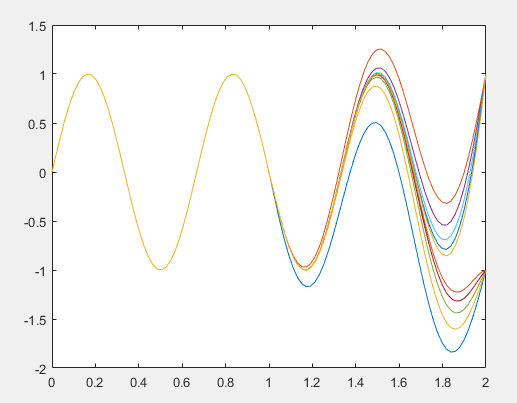
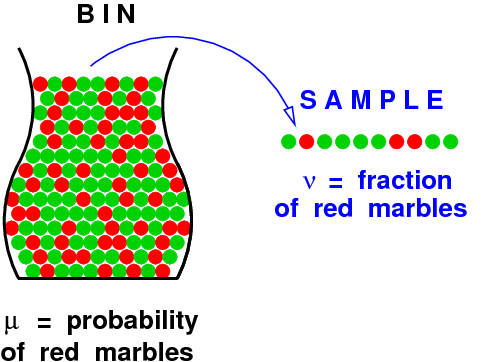


Figure 2 a series of target function Ln that has the same values in D but different outside of D

If we remain true to the notion of unknown target, we have a dilemma. The whole purpose of learning is to be able to predict the value of f on points that we haven't seen before. The quality of the learning will be determined by how close our prediction is to the true value. So regardless how g approach f it can agree or disagree with the target, depending on which f in the family of “fn” is turns out to be the true target.

# Generalizing using samples, example Bins

We will show that we can indeed infer something outside D using only D, but in a probabilistic way. What we infer may not be much compared to learning a full target function, but it will establish the principle that we can reach outside D. Once we establish that, we will take it to the general learning problem and pin down what we can and cannot learn. To this end we will start giving an example.

Consider a bin that contains red and green marbles. The proportion of red and green marbles in the bin is such that if we pick a marble at random, the probability that it will be red is µ and the probability that it will be green is 1 - µ. We assume that the value of µ is **unknown** to us ( remember f). If We pick a random sample of **N** independent marbles (with replacement) from this bin, and observe the fraction **v** of red marbles within the sample. What does the value of **v** tell us about the value of µ?

**Figure 3 Illustrative Figure**

One answer is that regardless of the colors of the N marbles that we picked, we still don't know the color of any marble that we didn't pick. We can get mostly green marbles in the sample while the bin has mostly red marbles. Although this is certainly **possible**, it is by no means **probable**.

The probability distribution of the random variable **v** in terms of the parameter µ is well understood, and when the sample size is big, **v** tends to be close to µ. To quantify the relationship between **v** and µ, we use a simple bound called the **Hoefding Inequality**. It states that for any sample size **N**,

1. N is the size of the training data
2. is the tolerance or the accuracy wanted

Remember µ is a constant unknown, but a constant none the less, so what makes the probability vary is v, also the probability is born regardless of µ or **v**. Here we see exactly the dynamic between the data and the tolerance, if I wanted height accuracy a.k.a. small, i need to have a large amount of data i.e. N must be big in order for the second right side to be close to zero.

|  |
| --- |
| *Attention :*  *The fact that the sample was randomly selected from the bin is the reason we are able to make any kind of statement about µ being close to* ***v****. If the sample was not randomly selected but picked in a particular way, we would lose the benefit of the probabilistic analysis and we would again be in the dark outside of the sample.* |

## Relation to the learning problem

We saw an example on how a sample can generalize an entire space, but in our example µ is an unknown constant but our f is an unknown function, but the two situation can be liked if we illustrate the procedures as follow:

1. The red marbles represent the instances (observations) that for **a given** h in H, such as h **different** of f
2. The green marbles represent the opposite i.e. the instances where f=h

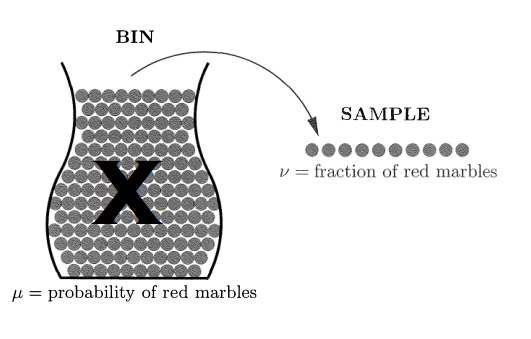
Here is when it gets a little bit complicated, in the bin we knew what the colors, we can see them, here we don’t, why? Because in our hypothesis **f** is **unknown**. But what we know are its values (or its characteristic) at a training set. We can remedy this be choosing the entire bin as the space X.

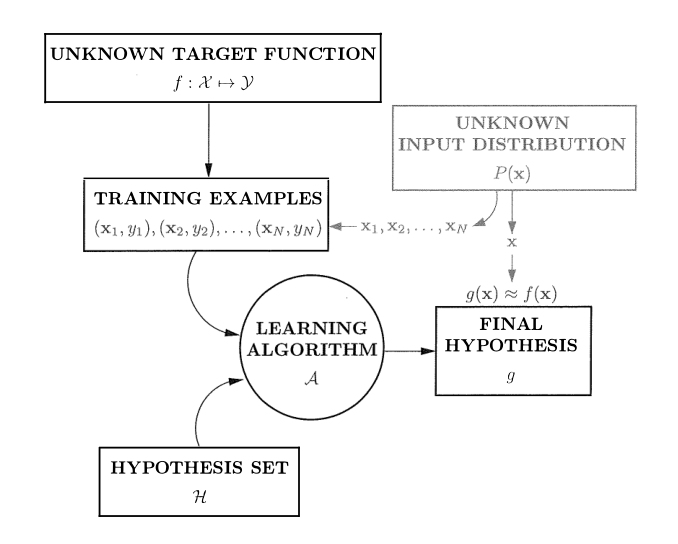
Figure 4 the marbles here are not colored here because we don't know the target function to be able to define them so µ is unknown

|  |
| --- |
| *Example:*  *We want to binary classify car images i.e. if a picture is car or not, the space X will be the entire car pictures ever taken since the taken of pictures of cars began of any model, type, shape or year. The sample will be our dataset (CIFAR-1000, CIFAR-100, etc.). We pick our learning model as Neural Network with backpropagation inception model, if we go back to our bin metaphor, we pick a picture x, the model give us a prediction .i.e. h(x), if for that particular picture, the target function and the h agree i.e. f(x)=h(x), we color it green else we color it red.* |

However, if we pick x at random according to some probability distribution P over the input space X, we know that x will be red with some probability, call it µ, and green with probability 1 - µ. Regardless of the value of µ, the space X now behaves like the bin.

The learning problem is now reduced to a bin problem, under the assumption that the inputs in D **are picked independently according to some distribution P** on X. Any P will translate to some µ in the equivalent bin. Since µ is allowed to be unknown, P can be unknown to us as. Now the by adding this information to the learning problem, it will be transform, to figure 5.

Figure 5 the learning problem with added probability



With this assumption, the Hoeffding Inequality (1) can be applied to the learning problem, allowing us to make a prediction outside of D.

**Using v to predict µ** tells us something about f, although it **doesn't tell us what f is**. What µ tells us is **the error rate h makes in approximating f**. If v happens to be **close to zero**, we can predict that **h will approximate f well** over the entire input space. If not, we are out of luck.

What we said so far was good for **“a given h”** a single h i.e. a single bin, now let us see if we can extend the bin equivalence to the case where we have multiple hypotheses in order to capture real learning.

To do that, we start by introducing more descriptive names for the different components that we will use. The error rate within the sample, which corresponds to v in the bin model, will be called ***the in-sample error***.

Or

We have made explicit the dependency of Ein on the particular h that we are considering. In the same way, we define the ***out-of-sample error.***

Which corresponds to µ in the bin model. The probability is based on the distribution P over X which is used to sample the data points x.

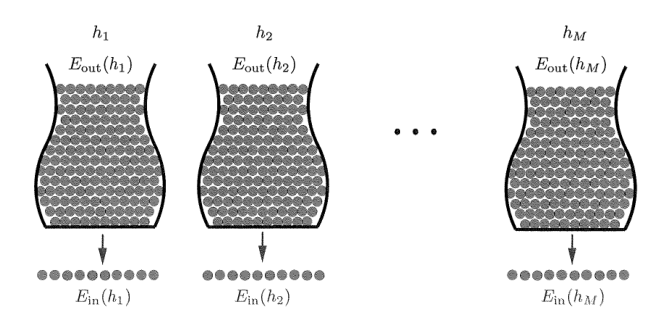
Substituting the new notation Ein for v and Eout for µ, the Hoeffding Inequality (1) can be rewritten as

Where:

1. N is the number of training examples.
2. The in-sample error Ein, just like v, is a random variable that depends on the sample.
3. The out-of-sample error Eout, just like µ, is unknown but not random.

Let us consider an entire hypothesis set H instead of just one hypothesis h, and assume for the moment that H has a finite number of hypotheses (we will see the case when we have an infinite number of hypothesis in theory of generalization).

Although the Hoeffding Inequality (2) still applies to each bin individually, the situation becomes more complicated when we consider all the bins simultaneously. So we need an equivalence that is true **for all** bins i.e. **for all** hm m={1,…,M}, because the hypothesis that we are interested in is g (the one the learning algorithm picks out) and g can be **any** hm.

 Figure 6 Illustration of multiple Bins with corresponding h distribution

To derive the equivalence needed we need the following property about probabilities:

1. Let A and B be two events if A🡪B then
2. Let Ti be a series of events then

The final hypothesis g based on D, i.e. after generating the data set. The statement we would like to make is

The hypothesis g is not fixed ahead of time before generating the data, because which hypothesis is selected to be g depends on the data. So, we cannot just plug in g for h in the Hoeffding inequality. But as stated earlier g can be **any** hm. So what we get is:

So probability becomes:

Mathematically, this is a “uniform” version of (2). We are trying to simultaneously approximate all Eout(hm)'s by the corresponding Ein(hm)'s. This allows the learning algorithm to choose any hypothesis based on Ein and expect that the corresponding Eout will uniformly follow suit, regardless of which hypothesis is chosen. The downside for uniform estimates is that the probability bound is a factor of M looser than the bound for a single hypothesis, and will only be meaningful if M is finite. We will improve on that in generalization theory.

# Feasibility of Learning

The question of whether D tells us anything outside of D that we didn't know before has two different answers. If we insist on a deterministic answer, which means that D tells us something certain about f outside of D, then the answer is no. If we accept a probabilistic answer, which means that D tells us something likely about f outside of D, then the answer is yes. If the only viable solution (dictated by your customer, environment, …) requires to be deterministic, do not use machine learning to solve your problem.

If we adopt a probabilistic view, the only assumption we make in the probabilistic framework is that the examples in D are generated independently. We don't insist on using any particular probability distribution, or even on knowing what distribution is used. However, whatever distribution we use for generating the examples, we must also use when we evaluate how well g approximates f. That's what makes the Hoeffding Inequality applicable. Of course this ideal situation may not always happen in practice, and some variations of it have been explored in the literature. In case of synthetic data, we need to be careful to not induce any bias or observation dependencies, when generating the dataset. Using multiple distribution can in fact give bad results. In case of real but incomplete data like imputation data, replacing unknown values by the mean or any other value can result in dependency issues, a solution is to solve the inference problem to figure out the probability distribution over the dataset then invers it to generate samples. So **GOOD** data is a **MUST.**

Until now we just figure out the first part that is generalizing using a sample, what we get from a probabilistic analysis is , but we need more than that, we need The Hoeffding Inequality (3) addresses the first part only. The second question is answered after we run the learning algorithm on the actual data and see how small we can get Ein to be. A lot of people ignore the first part and dive directly to the second, maybe there algorithm will achieve great performance (i.e. getting ) but they launch it without any guarantee that it will make the same performance. On the other hand we can only be sure of but can never achieve this is an acceptable procedure (Financial forecasting is an example where market unpredictability makes it impossible to get a forecast that has anywhere near zero error. All we hope for is a forecast that gets it right more often than not. If we get that, our bets will win in the long run. This means that a hypothesis that has Ein (g) somewhat below 0.5 will work, provided of course that Eout(g) is close enough to Ein(g)).