Notes on Foundations of Machine Learning by M. Mohri

compiled by D. Gueorguiev 3/16/2025

# The Probabilistically Asymptotically Correct (PAC) Learning Model

- the set of all possible *examples* or *instances*. also is referred to as the *input space*.

– the set of all possible labels / target values. When we have *binary classification*.

A *concept* is a mapping from to . Since , we can identify with the subset of over which it takes the value 1.

A *concept class* is a set of concepts.

Let us assume that the examples are independently and identically distributed (i.i.d.) according to some fixed but unknown distribution . The learner considers a fixed set of possible concepts called hypothesis set, which may not coincide with . The learner receives a sample drawn i.i.d. according to as well as the labels set which are based on a specific target concept to learn. The learner’s task is to use the labeled sample to select a hypothesis that has a small *generalization error* with respect to the concept . The generalization error of a hypothesis , also referred to as the *true error* of is denoted with and is defined as follows:

**Definition** *Generalization error*

Given a hypothesis , a target concept and an underlying distribution , the generalization error or risk of is defined by

(1)

where is the indicator function of the event .

The generalization error of a hypothesis is not directly accessible to the learner since both the distribution and the target concept are unknown. However, the learner can measure the *empirical error* of a hypothesis on the labeled sample .

**Definition** *Empirical error*

Given a hypothesis , a target concept , and a sample , the empirical error *or* empirical risk of is defined by

(2)

Thus, the empirical error of is its average error over the sample , while the generalization error is the expected error based on the distribution .

**Statement**:

for fixed , the expectation of the empirical error based on an i.i.d. sample is equal to the generalization error:

(3)

Proof:

By the linearity of the expectation and the fact that the sample is drawn i.i.d., we can write

for any in sample . Thus,

The following introduces the *Probabilistically Asymptotically Correct* (PAC) learning framework. We denote by an upper bound on the cost of the computational representation of any element and by the maximal cost of the computational representation of . For example, , for which the cost of an array-based representation would be in .

**Definition** *PAC-learning*

A concept class is said to be PAC-learnable if there exists an algorithm and a polynomial function such that for any and , for all distributions on and for any target concept , the following holds for any sample size :

(4)

If further runs in , then is said to be efficiently PAC-learnable. When such algorithm exists, it is called a PAC-learning algorithm for .

A concept class is thus PAC-learnable if the hypothesis returned by the algorithm after observing a number of points polynomial in and is *asymptotically correct* (error at most ) with high probability (at least ).

is used to define the *confidence* and the *accuracy* . Note that if the running time of the algorithm is polynomial in and , then the sample size must also be polynomial if the full sample is received by the algorithm.

**Notes**:

i ) The PAC framework represents a distribution-free model – no assumption is made about the distribution from which the examples are drawn.

ii ) The training sample and test examples used to define the error are drawn according to the same distribution D. This assumption is necessary for the generalization to be possible.

iii ) The PAC framework deals with the question of learnability of a concept class and not a particular concept . The concept class is known to the algorithm, but the target concept is unknown.

**Example** Learning axis-aligned rectangles

A diagram of a diagram

Description automatically generatedFigure 1: target concept and possible hypothesis .

Consider the case where the set of instances are points in the plane, , and the concept class is the set of all axis-aligned rectangles lying in . Thus, each concept is the set of points inside a particular axis-aligned rectangle. The learning problem consists of determining with small error a target axis-aligned rectangle using the labeled training sample. We will show that the concept class of axis-aligned rectangles is PAC-learnable.

On Figure 1 corresponds to the set of *false negatives* while corresponds to the set of *false positives*.

To show that the concept class is PAC-learnable, we describe a simple PAC-learning algorithm . Given a labeled sample , the algorithm consists of returning the tightest axis-aligned rectangle containing the points labeled with 1.

A diagram of a rectangle with red and blue dots

Description automatically generatedFigure 2: the hypothesis returned by the algorithm

By definition, does not produce any false positive, since its points must be included in the target concept R. Thus, the error region of is included in .

Let be a target concept. Fix . Let denote the probability mass of the region defined by – this is the probability that a point randomly drawn according to falls within . Since errors made by our algorithm can be due only to points falling inside , we can assume that ; otherwise, the error of is less than or equal to regardless of the training sample received.

Now, since , we can define four rectangular regions , and along the sides of , each with probability at least . These regions can be constructed by starting with the empty rectangle along a side and increasing its size until its distribution mass is at least . Figure 3 depicts those regions.

A diagram of a rectangle with red and blue dots

Description automatically generated

Figure 3: the side regions , , ,

Observe that if meets each of these four regions, then, because it is a rectangle, it will have one side in each of those four regions (geometric argument). Its error area, which is the part of that it does not cover, is thus included in these regions and cannot have probability mass more than .

# References

[1] [Foundations of Machine Learning, Mehryar Mohri, Afshin Rostamizadeh, Ameet Talwalkar, 2012](https://github.com/dimitarpg13/statistical_learning_and_kernel_methods/blob/main/literature/books/Foundations_of_Machine_Learning_Mohri_2012.pdf)

[2]

# Appendix

## Cross-validation

in practice the amount of labeled data available is often too small to set aside a validation sample since that would leave an insufficient amount of training data. Instead, a widely adopted method known as n-fold cross-validation is used to exploit the labeled data both for model selection (selection of the free parameters of the algorithm) and for training.

Let denote the vector of free parameters of the algorithm. For a fixed value of , the method consists of first randomly partitioning a given sample of labeled examples into subsamples or *folds*. The -th fold is thus a labeled sample of size . Then, for any , the learning algorithm is trained on all but the th fold to generate a hypothesis , and the performance of is tested on the -th fold. The parameter value is evaluated based on the average error of the hypotheses , which is called the *cross-validation error*. This quantity is denoted by and defined by

(A1.1)

In (A1.1) represents the loss function which measures the difference or loss between the predicted label and a true label. Denoting the set of all labels as and the set of possible predictions as , a loss function is a mapping . In most cases and the loss function is bounded, but these conditions do not always hold. Common examples of loss function are:

the *zero-one* (or misclassification) loss function defined over by

the squared loss function defined over by where is typically a bounded interval.

Then the expression represents the error of on the -th hold. The folds are generally chosen to have equal size, that is for all .

Question: How should be chosen?

Each training sample used in the -fold cross validation has size . When is large the size of the training sample is close to , the size of the full sample, but the training samples are quite similar; thus, for large we tend to have small bias but a large variance. In contrast, smaller values of lead to more diverse training samples but their size is significantly less than , thus the method tends to have a smaller variance but a larger bias.

-fold cross validation is used as follows in model selection. The full labeled data is first split into a training and a test sample. The training sample of size is then used to compute the -fold cross-validation error for a small number of possible values of . is next set to the value for which is smallest and the algorithm is trained with the parameter setting over the full training sample of size . Its performance is evaluated on the test sample.

Special case: *leave-one-out cross-validation*

This is the case when since at each iteration exactly one instance is left out of the training sample.

The average leave-one-out error is an approximately unbiased estimate of the average error of an algorithm and can be used to derive simple guarantees for some algorithms. In general, the leave-one-out error is very costly to compute since it requires training times on samples of size , but for some algorithms it admits a very efficient computation.

Note: *n-fold cross validation for performance evaluation*

-fold cross validation is also used for performance evaluation. In that case, for a fixed parameter setting , the full labeled sample is divided into random folds with no distinction between training and test samples. The performance reported is the n-fold cross-validation on the full sample as well as the standard deviation of errors measured on each fold.