MATH 6397 - Mathematics of Data Science

Instructor: Demetrio Labate

April 4, 2023

Course Outline

1. Mathematics of machine learning
    1.1 Statistical learning theory

References:

- *Foundations of Data Science*, by Blum, Hopcroft and Kannan
- *Foundations of Machine Learning*, by Mohri, Rostamizadeh and Talwalkar

Statistical Learning Theory

Machine Learning

Machine Learning originated in the computer science community and can be roughly described as the field of study concerned with the development of *methods for learning good models from data.*

It is centered around the concepts of **data, learning** and **models**.

The main goal of machine learning is to compute a model that can predict unseen data.

A *good model* is as good as its ability to formulate an accurate prediction.

Machine Learning

Standard machine learning tasks:

1. **Classification.** This is the problem of assigning a category to each item.
2. **Regression.** This is the problem of predicting a real value for each item.
3. **Ranking.** This is the problem of learning to order items according to some criterion.
4. **Clustering.** This is the problem of partitioning a set of items into homogeneous subsets.
5. **Dimensionality reduction (or manifold learning).** This problem consists of transforming an initial representation of items into a lower-dimensional representation while preserving some properties of the initial representation.

Machine Learning

There are several machine learning scenarios which differ in the types of training data available to the learner, the order and method by which training data is received, and the test data used to evaluate the learning algorithm.

1. **Supervised learning.** The learner receives a set of labeled examples as training data and makes predictions for all unseen points. This is the most common scenario associated with classification, regression, and ranking problems.
2. **Unsupervised learning.** The learner exclusively receives unlabeled training data, and makes predictions for all unseen points. Clustering and dimensionality reduction are example of unsupervised learning problems.
3. **Semi-supervised learning.** The learner receives a training sample consisting of both labeled and unlabeled data, and makes predictions for all unseen points. It may be useful when unlabeled data is easily accessible but labels are expensive to obtain. Classification, regression, or ranking tasks, can be framed as instances of semi-supervised learning.

Machine Learning

4. **Transductive inference.** As in the semi-supervised scenario, the learner receives a labeled training sample along with a set of unlabeled test points. The objective is to predict labels only for these particular test points.
5. **On-line learning.** This scenario involves multiple rounds where training and testing phases are intermixed. At each round, the learner receives an unlabeled training point, makes a prediction, receives the true label, and incurs a loss. The objective is to minimize the cumulative loss over all rounds.
6. **Reinforcement learning.** The training and testing phases are intermixed. To collect information, the learner actively interacts with the environment and receives an immediate reward for each action. The object of the learner is to maximize his reward over a course of actions and iterations with the environment.
7. **Active learning.** The learner interactively collects training examples, typically by querying an oracle to request labels for new points.

Machine Learning

To introduce some basic concepts, let us consider the problem of recognizing handwritten digits in the MNIST database.

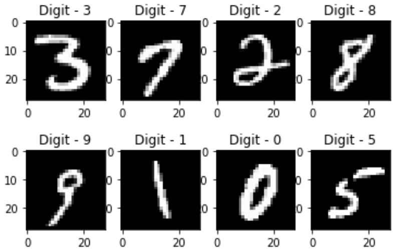


Figure: Handwritten digits and corresponding labels, taken from the MNIST database.

Each **example** is a 28×28 pixel image and so can be represented as a vector $x \in X = \mathbb{R}^{784}$. To each example, is associated a **label**[1] y that identifies the digit value, $y \in Y = \{0, 1, \ldots, 9\}$.

[1]The *label* has other names, including target, response variable, and annotation.

Machine Learning

The goal of machine learning is to build a **predictor** or **model** that maps an input $x \in X$ to an output $y \in Y$

$$x \in X \quad \mapsto \quad y \in Y$$

In the supervised learning setting, we use a **training set** consisting of example-label pairs

$$S = \{(x_1, y_1), \ldots, (x_N, y_N)\} \subset X \times Y$$

to tune the parameters of the predictor.

Once the model is trained, it can be used to predict the labels of new images $x \in X$ which are said to comprise a **test set.**

For instance, the MNIST database contains 60,000 training images and 10,000 testing images.

Machine Learning

Remark. In many applications, the original input variables are **preprocessed** to transform them in some new space of variables where - it is hoped - the pattern recognition problem will be easier to solve.

For instance, the images in the MNIST database are typically translated and scaled so that each digit is contained within a box of fixed size.

Preprocessing can also include transformations that reduce dimensionality such as low-dimensional approximations using principal components as well as mapping into high-dimensional representations.

This pre-processing stage is sometimes called **feature extraction** and **feature map** is the corresponding transformation.
As a result the input vectors $x \in X$ of a machine learning algorithm are often called **feature vectors.**

Machine Learning

The predictor can be either a **function** or a **probabilistic model**.

- Case (1): the predictor is a function

  $$f : x \in X \mapsto y = f(x) \in Y$$

  For example, we can have $f : \mathbb{R}^D \to \mathbb{R}$ of the form

  $$f(x) = \theta_1^t x + \theta_0$$

  where $\theta = (\theta_0, \theta_1) \in \mathbb{R} \times \mathbb{R}^D$ are the **parameters** of the predictor.
- Case (2): the predictor is a probabilistic model. For instance, the predictor is a probability density function with finitely many **parameters** such as the normal distribution.

Machine Learning

In general, we can identify three algorithmic phases in the supervised learning process.

1. **Model selection.** As part of the learning process, we need to make high-level decisions about the structure of the predictor, e.g., the predictor is a polynomial function of a certain degree or the predictor is a normal distribution.
2. **Training** (also called **parameter estimation**). During this phase, we adjust the parameters of the predictor based on training data and, for that, we need a measure of quality to control the performance of the predictor. To perform this task, there are two main strategies depending on the predictor being a function or a probabilistic model: **empirical risk minimization** and **maximum likelihood estimation,** resp.
3. **Prediction or inference.** During this phase, the predictor is applied to unseen data, i.e., the test data, to generate an outcome.

Machine Learning

In the supervised learning process described above, the parameters of the model are determined using the training data.

The performance of the model is assessed on the test data.

In general, there is no guarantee that the model will perform as well on the test data as on the training data
(more about this topic below)

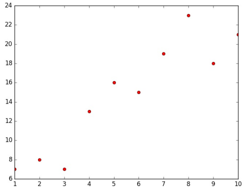
The ability of the trained model to categorize correctly new examples (not part of the training set) is called **generalization.**

Machine Learning

The regression problem is useful to illustrate some fundamental concepts.

Example: Polynomial Curve Fitting
Suppose we are given a set of data pairs

$$S_N = \{(x_1, y_1), \dots, (x_N, y_N)\} \subset \mathbb{R}^d \times \mathbb{R}$$



Our goal is to exploit this (training) set in order to find a predictor $f(\cdot, \theta)$, parametrized by θ, that we can use to compute the target value $y \in \mathbb{R}$ for some new input variable $x \in \mathbb{R}^d$.

In practice, after choosing an appropriate class of functions, we have to find the parameter θ^* giving the best fit of f to the data so that we can estimate y as $\hat{y} = f(x, \theta^*)$.

In the polynomial regression problem, the predictor $f(\cdot, \theta)$ is chosen to be a polynomial function.

Machine Learning

We use a polynomial of order M to fit the data

$$f(x,\theta) = \theta_0 + \theta_1^t x + \cdots + \theta_M^t x^m = \sum_{j=0}^{M} \theta_j^t x^j,$$

where $\theta = (\theta_0, \ldots, \theta_M)$ and $\theta_0 \in \mathbb{R}$, $\theta_j \in \mathbb{R}^d$, $j = 1, \ldots, M$. Note that, although the polynomial is not a linear function in general, it is a linear function of the parameter θ.

Predictor functions $f(\cdot, \theta)$ that are linear with respect to the parameter θ are called **linear models**.

The parameter θ^* giving the best fit of f to the data is determined by the introduction of an appropriate **loss (or error) function**

$$L(y_j, f(x_j, \theta))$$

whose output is a non-negative number, *the loss*, measuring the error made in this particular prediction.

Machine Learning

Empirical risk minimization

We assume that the example pairs $S_N = \{(x_1, y_1), \dots, (x_N, y_N)\}$ are independent and identically distributed (*i.i.d.*).
This means that any two data points (x_i, y_i) and (x_j, y_j) are statistically independent of each other and are drawn from the same (unknown) distribution.

This assumption implies that (for a large sa,ple size N) the empirical mean is a good estimate of the population mean. Hence we define the following concept.

Definition (Empirical Risk)

The **empirical risk** is the average loss on the training data

$$R_{emp}(f, S_N) = \frac{1}{N} \sum_{n=1}^{N} L(y_n, f(x_n; \theta)).$$

Machine Learning

If we set the degree of the polynomial as $M = 1$, then we look for a predictor function of the form

$$f(x; \theta_0, \theta_1) = \theta_1^t x + \theta_0, \quad x \in \mathbb{R}^d, \theta_1 \in \mathbb{R}^d, \theta_0 \in \mathbb{R}.$$

Remark

We can map any $x = (x_1, \dots, x_d)^t \in \mathbb{R}^d$ to the vector $\tilde{x} = (1, x_1, \dots, x_d)^t \in \mathbb{R}^{d+1}$ and similarly any $\theta = (\theta_1, \dots, \theta_d)^t \in \mathbb{R}^d$ to $\tilde{\theta} = (\theta_0, \theta_1, \dots, \theta_d)^t \in \mathbb{R}^{d+1}$.

By the remark, we can redefine the affine function f above as the linear function $\tilde{f} : \mathbb{R}^{d+1} \to \mathbb{R}$ given by

$$\tilde{f}(\tilde{x}, \tilde{\theta}) = \tilde{\theta}^t \tilde{x}.$$

Clearly, using the same observation, we can absorbe the constant term θ_0 into $\tilde{\theta}$ for any polynomial function f.

Machine Learning

To find the best parameter $\tilde{\theta}$ of the predictor $\tilde{f}(\cdot, \tilde{\theta})$ for the given training data S_N, we can use the **squared loss** function

$$L(y, \tilde{f}(x, \tilde{\theta})) = (y - \tilde{f}(\tilde{x}; \tilde{\theta}))^2.$$

Using this loss function, the empirical risk becomes

$$R_{emp}(\tilde{f}, S_N) = \frac{1}{N}\sum_{n=1}^{N}(y_n - \tilde{f}(\tilde{x}_n, \tilde{\theta}))^2 = \frac{1}{N}\sum_{n=1}^{N}(y_n - \tilde{\theta}^t\tilde{x})^2.$$

To minimize the empirical risk we solve

$$\min_{\tilde{\theta}\in\mathbb{R}^{D+1}} \frac{1}{N}\sum_{n=1}^{N}(y_n - \tilde{\theta}^t\tilde{x})^2 = \min_{\tilde{\theta}\in\mathbb{R}^{D+1}} \frac{1}{N}\|Y - \tilde{X}\tilde{\theta}\|^2$$

where Y is the vector containing the labels y_n as entries and $\tilde{X}$ containing the data points $\tilde{x}_n$ as columns. This minimization problem is known as the **least-square linear regression problem**.

Machine Learning

We seek to find a predictor that performs well on unseen data.

That is, we want to find a predictor function $f(\cdot, \theta)$ that minimizes the **expected risk**

$$R_{true}(f) = \mathbb{E}_{x,y}[L(y, f(x))]$$

where y is the label of x and $f(x)$ is the prediction.
Here the expectation $E_{x,y}$ is taken over the infinite set of all possible data and labels.

The notion of expected risk leads to a number of practical questions, such as:

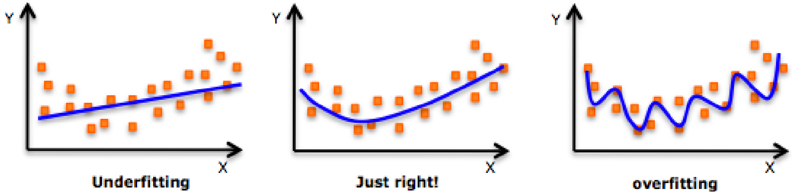
1. How to estimate the expected risk?
2. How to control the expected risk from the empirical risk?

Machine Learning

Minimization of the empirical risk is no guarantee that the expected risk is minimized.

Empirical risk minimization may lead to **overfitting**.

This is the situation where the predictor $f(\cdot, \theta)$ fits closely the training data but does not generalize well on new data, that is, $R_{emp}(f)$ underestimates $R_{true}(f)$.



Machine Learning

A useful strategy to avoid overfitting is **regularization** consisting in finding a compromise between accurate solution of empirical risk minimization and complexity of the model.

In other words, regularization discourages complex or extreme solutions to an optimization problem in favor of simpler ones.

Example: Regularized linear least squares.
This regularization strategy adds a penalty term involving the parameter θ:

$$min_{\tilde{\theta}\in\mathbb{R}^D} \frac{1}{N}\|Y - \tilde{X}\tilde{\theta}\|^2 + \lambda\|\tilde{\theta}\|^2.$$

The term $\|\theta\|^2$ is a **regularizer** and λ is the regularization parameter.

The effect of the regularization is to force the solution to be *sparse* in some way which might reflect some prior knowledge about the problem.

Machine Learning

Predictor as a probability model.
An alternative point view uses a probability model rather than a function as predictor

In this setting, we assume that data are random variables associated with a probability density function $p(x; \theta)$, parametrized by θ, and we want to determine the parameter vector θ that best fits the data.

We define the **negative log-likelihood** by

$$\mathcal{L}_x(\theta) = -\log p(x|\theta).$$

In this expression, θ is the variable (i.e., the quantity we want do find, for the given data) and x is fixed.

To find the value of the parameter vector θ that best fit the data, we **maximize the likelihood**.
Equivalently, we minimize $\mathcal{L}_x(\theta)$ with respect to θ.

Machine Learning

Example: Gaussian distribution.

Here we assume that we can explain data uncertainty using a Gaussian probability model, i.e., the uncertainty is a Gaussian noise with zero mean and fixed variance σ^2.

In addition, we also assume a linear model $\theta^T x_n$ for prediction.

Hence, for any observation (x_n, y_n)

$$p(y_n|x_n, \theta) = \mathcal{N}(y_n|\theta^T x_n, \sigma^2).$$

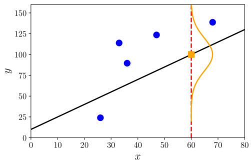


Figure: The uncertainty of the data is described using a Gaussian model.

Machine Learning

Under the assumption that the data $(x_1, y_1), \ldots, (x_N, y_N)$ that are i.i.d., the likelihood of the whole data set factorizes into a product of the likelihoods of each individual example

$$P(Y|X,\theta) = \Pi_{n=1}^{N} p(y_n|x_n,\theta),$$

where

$$Y = \{y_1, \ldots, y_N\}, \quad X = \{x_1, \ldots, x_N\}$$

and

$$p(y_n|x_n,\theta) = \mathcal{N}(y_n|\theta^T x_n, \sigma^2) \quad \text{for each } n$$

Machine Learning

Hence, we compute the negative log-likelihood as

$$
\begin{aligned}
\mathcal{L}(\theta) &= -\sum_{n=1}^{N} \log\ p(y_n|x_n,\theta) \\
&= -\sum_{n=1}^{N} \log\ N(y_n|\theta^T x_n,\sigma^2) \\
&= -\sum_{n=1}^{N} \log\ \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{y_n-\theta^T x_n}{2\sigma^2}\right) \\
&= \frac{1}{2\sigma^2}\sum_{n=1}^{N}(y_n-\theta^T x_n)^2 - \sum_{n=1}^{N}\log\frac{1}{\sqrt{2\pi\sigma^2}}.
\end{aligned}
$$

We solve the maximum likelihood estimation by minimizing $\mathcal{L}(\theta)$. The solution is equivalent to the least-square regression problem

$$
\min_{\theta}\sum_{n=1}^{N}(y_n-\theta^T x_n)^2
$$

The PAC Learning Framework

Some fundamental questions arise when designing and analyzing algorithms that learn from examples:

- What can be learned efficiently?
- How many examples are needed to learn successfully?

The **Probably Approximately Correct (PAC)** learning framework, introduced by Valiant in 1984, defines the class of learnable **concepts** in terms of

1. **number of samples** needed to achieve an approximate solution;
2. **sample complexity**;
3. **computational complexity** of the learning algorithm.

This approach combines statistical pattern recognition, decision theory and computational complexity theory to came up with a notion of learning problems that are **feasible** in the sense that there is a polynomial time algorithm that solves them in analogy with the class of feasible problems in standard complexity theory.

The PAC Learning Framework

Notation:

- The **input space** X containing all possible examples or instances. For instance, $X = \mathbb{R}^2$
- The **target space** or space of **labels** Y. For instance, in the case of **binary classification**, $Y = \{0, 1\}$.
- A **concept** is a mapping $c : X \mapsto Y$. A **concept class** $\mathcal{C}$ is a set of concepts we may wish to learn. For instance, $\mathcal{C}$ could be the set of solid triangles in the plane.
- The **hypothesis space** H is the set of all concepts considered during the learning process. This space may or may not coincide with $\mathcal{C}$.
- A sample $S \subset X$ is a set of instances drawn i.i.d. according to some unknown distribution $\mathcal{D}$.

The PAC Learning Framework

The **learning problem** is formulated as follows.

The learner receives a sample $S = \{x_1, \ldots, x_m\}$ (i.i.d. according to an unknown distribution $\mathcal{D}$) as well as the corresponding labels $y_i = c(x_i)$, $i = 1, \ldots, m$, which are based on a specific concept $c \in \mathcal{C}$ to learn.

The task is then to use the sample S and the corresponding labels to select a hypothesis h_S in an appropriate hypothesis space H that has a small **generalization error** with respect to $c \in \mathcal{C}$.

Definition

Given a hypothesis $h \in H$, a target concept $c \in \mathcal{C}$ and an underlying distribution $\mathcal{D}$, the **generalization error** or **risk** of h is defined by

$$\mathcal{R}(h) = P_{x\sim\mathcal{D}}(h(x) \neq c(x)) = \underset{x\sim\mathcal{D}}{E}[1_{h(x)\neq c(x)}]$$

The PAC Learning Framework

Note. In the basic PAC model, the input space X is assumed to be $\{0,1\}^n$, that is, the set of all possible assignments to n Boolean variables or attributes; the concepts and hypotheses are subsets of $\{0,1\}^n$.
The error of a hypothesis h with respect to a fixed target concept c is then defined by

$$error(h) = \sum_{x \in h \Delta c} \mathcal{D}(x)$$

where Δ is the symmetric difference and $\mathcal{D}$ is the probability distribution defined on the instance space $\{0,1\}^n$.
Equivalently, $error(h)$ is the probability that h and c will disagree on an instance drawn randomly according to $\mathcal{D}$:

$$error(h) = P_{x \sim \mathcal{D}}(h(x) \neq c(x))$$

The PAC Learning Framework

Since $\mathcal{D}$ and c are unknown, the generalization error of a hypothesis is not directly accessible to the learner.

However, the learner can measure the empirical error of a hypothesis on the labeled samples.

Definition

Given a hypothesis $h \in H$ and a sample $S = (x_1, \ldots, x_m)$ of size m with the corresponding labels $c(x_i)$, $i = 1, \ldots, m$, the **empirical error** or **empirical risk** of h is defined by

$$\hat{\mathcal{R}}_S(h) = \frac{1}{m} \sum_{i=1}^{m} 1_{h(x_i) \neq c(x_i)}$$

The PAC Learning Framework

For a fixed $h \in H$, the expectation of the empirical error based on an i.i.d. sample S of size m is equal to the generalization error:

$$\underset{S \sim \mathcal{D}^m}{E}[\hat{\mathcal{R}}_S(h)] = \mathcal{R}(h)$$

That is, empirical error is an uubiased estimate of the generalization error.

Proof: using the fact that the sample is i.i.d., we have

$$\underset{S \sim \mathcal{D}^m}{E}[\hat{\mathcal{R}}_S(h)] = \frac{1}{m} \sum_{i=1}^{m} \underset{x_i \sim \mathcal{D}}{E}[1_{h(x_i) \neq c(x_i)}] = \frac{1}{m} \sum_{i=1}^{m} \underset{x \sim \mathcal{D}}{E}[1_{h(x) \neq c(x)}],$$

which holds for any x in S. Thus

$$\underset{S \sim \mathcal{D}^m}{E}[\hat{\mathcal{R}}_S(h)] = \underset{x \sim \mathcal{D}}{E}[1_{h(x) \neq c(x)}] = \mathcal{R}(h).$$

Note: This observation holds for a fixed h.

The PAC Learning Framework

We now define PAC learning framework.

Let n be a number such that the computational cost of representing any element $x \in X$ is at most $O(n)$ and $size(c)$ be the maximal cost of the computational representation of $c \in \mathcal{C}$. For example, x may be a vector in $\mathbb{R}^n$, for which the cost of an array-based representation would be in $O(n)$.

Definition

A concept class $\mathcal{C}$ is said to be **PAC-learnable** if there exists an algorithm $\mathcal{A}$ and a polynomial function q such that, for any $\epsilon > 0$ and $\delta > 0$, for all distributions $\mathcal{D}$ on X, and for any target concept $c \in \mathcal{C}$, the following holds for any sample size $m > q(\frac{1}{\epsilon}, \frac{1}{\delta}, n, size(c))$:

$$\underset{S\sim\mathcal{D}^m}{P}(\mathcal{R}(h_S) \leq \epsilon) \geq 1 - \delta,$$

where h_S is the hypothesis returned by algorithm $\mathcal{A}$ after receiving a labeled sample S.

The PAC Learning Framework

A concept class $\mathcal{C}$ is thus PAC-learnable if the hypothesis returned by the algorithm after observing a sufficiently many samples (polynomial in $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$) is approximately correct (generalization error is at most ϵ) with high probability (at least $1 - \delta$).

$1 - \delta$ is the **confidence** of the estimate
$1 - \epsilon$ is the **accuracy** of the hypothesis

If the algorithm $\mathcal{A}$ runs in $q(\frac{1}{\epsilon}, \frac{1}{\delta}, n, size(c))$, then $\mathcal{C}$ is said to be **efficiently PAC-learnable** and the algorithm is called a PAC-learning algorithm for $\mathcal{C}$.

Remarks.

- The PAC framework is distribution-free: no assumption is made about the distribution from which examples are drawn.
- The training sample and the test examples used to define the error are drawn according to the same distribution $\mathcal{D}$. This is a necessary assumption for generalization to be possible.

The PAC Learning Framework

Example: Learning axis-aligned rectangles.
Consider the case where the set of instances are points in the plane $X = \mathbb{R}^2$ and the concept class $\mathcal{C}$ is the set of all axis-aligned rectangles lying in $\mathbb{R}^2$:

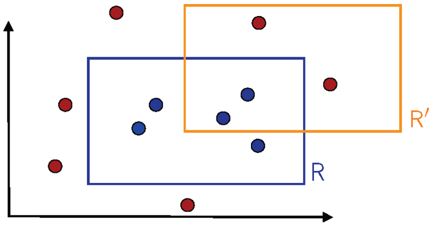
- Input space: $X = \mathbb{R}^2$
- Target space: $Y = \{-1, +1\}$
- Concept class: $\mathcal{C}$ = axis-aligned rectangles

Hence, each concept $c \in \mathcal{C}$ 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.

Claim: concept class $\mathcal{C}$ of axis-aligned rectangles is PAC-learnable

The PAC Learning Framework



Let $R \in \mathcal{C}$ be a target axis-aligned rectangle.
If only a sample S is observed, how do we guess $R \in \mathcal{C}$?

Suppose R' is a hypothesis. The (generalization) error regions of R' are formed by the area within R but outside R' (false negatives) and the area within R' but outside R (false positives).

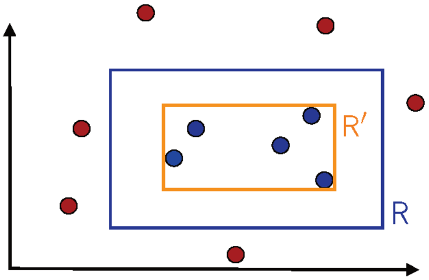
False negatives: points that are labeled as -1 by R', which are in fact labeled with 1.
False positives: that is, points labeled $+1$ by R' which are in fact labeled with -1.

The PAC Learning Framework

We propose a simple PAC-learning algorithm $\mathcal{A}$:

Given a labeled sample S, $\mathcal{A}$ returns the tightest axis-aligned rectangle $R' = R_S$ containing the points labeled with 1.

By construction, R_S is a subset of R (that is, $h_s \subset c$) and does not produce any false positives.
Thus, the error region of R_S is included in R.

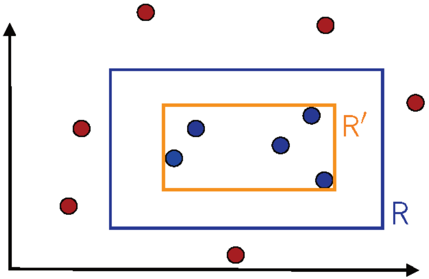


If one takes more instances, they may occupy the region in $R \setminus R_s$ leading to a smaller generalization error.

The PAC Learning Framework

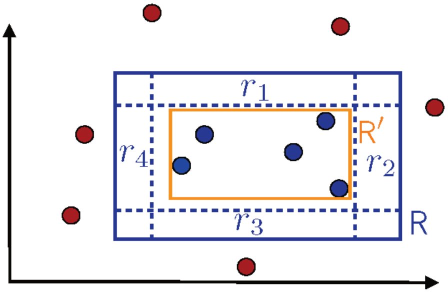
Let $P(R)$ denote the probability mass of the region defined by R, that is the probability that a point randomly drawn according to $\mathcal{D}$ falls within R.

Since errors made by our algorithm can be due only to points falling inside R, we can assume that $P(R) > \epsilon$, where $\epsilon > 0$. Otherwise, the error of $R' = R_S$ is less than or equal to ϵ regardless of the training sample S received.



The PAC Learning Framework

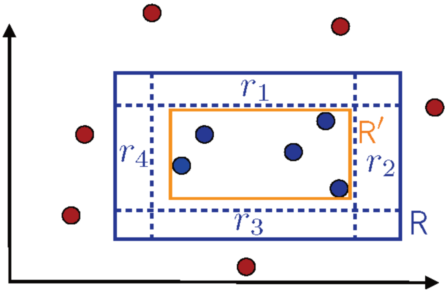
We can define four rectangular regions r_1, r_2, r_3, and r_4 along the sides of R, each with probability at least $\epsilon/4$.
These regions can be constructed by starting with the full rectangle R and then decreasing the size by moving one side as much as possible while keeping a distribution mass of at least $\epsilon/4$.



The PAC Learning Framework

If $R' = R_S$ meets all of these four regions r_i, then, because it is a rectangle, it will have one side in each of these regions.

Its error area, which is the part of R that it does not cover, is thus included in the union of the regions r_i and cannot have probability mass more than ϵ.
By contraposition, if $\mathcal{R}(R_S) > \epsilon$, then R must miss at least one of the regions r_i.



The PAC Learning Framework

By contraposition, if $\mathcal{R}(R_S) > \epsilon$, then R must miss at least one of the regions r_i. In this case

$$\begin{aligned}
\underset{S\sim\mathcal{D}^m}{P}(\mathcal{R}(R_S) > \epsilon) &\leq \underset{S\sim\mathcal{D}^m}{P}(\cup_{i=1}^{4}\{R_s \cap r_i = \varnothing\}) \\
&\leq \sum_{i=1}^{4} \underset{S\sim\mathcal{D}^m}{P}(\{R_s \cap r_i = \varnothing\}) \\
&\leq 4\,(1-\tfrac{\epsilon}{4})^m \quad (\text{since } P(r_i) \geq \tfrac{\epsilon}{4}) \\
&\leq 4\,e^{-m\frac{\epsilon}{4}}
\end{aligned}$$

where, in the last step, we used $1 - x \leq e^{-x}$.
For any $\delta > 0$, to ensure $\underset{S\sim\mathcal{D}^m}{P}(\mathcal{R}(R_S) > \epsilon) \leq \delta$, we can impose

$$4\,e^{-m\frac{\epsilon}{4}} \leq \delta \iff m \geq \tfrac{4}{\epsilon}\log\tfrac{4}{\delta} \quad (\text{sample complexity})$$

Thus, if the sample size m is greater than $\frac{4}{\epsilon}\log\frac{4}{\delta}$, we have

$$\underset{S\sim\mathcal{D}^m}{P}(\mathcal{R}(R_S) > \epsilon) \leq \delta$$

The PAC Learning Framework

In addition, we observe that the computational cost of the representation of points in $\mathbb{R}^2$ and axis-aligned rectangles, which can be defined by their four corners, is constant.

This proves that the concept class of axis-aligned rectangles is PAC-learnable and that the sample complexity of PAC-learning axis-aligned rectangles is of order $O(\frac{1}{\epsilon} \log \frac{1}{\delta})$

The PAC Learning Framework

Here is the generalization of the above observation with the corresponding sample complexity.

Theorem

Consider the input space $X = \mathbb{R}^d$ and the concept class $\mathcal{C}$ of all faced-aligned closed hypercubes in $\mathbb{R}^d$. That is each concept $c \in \mathcal{C}$ is the set of points inside a particular face-aligned hypercube. Consider the algorithm that, given a labeled sample S returns the tightest face-aligned hypercubeV_s consisting of points labeled with $+1$. Then

$$P(\mathcal{R}(V_S) \leq \epsilon) \geq 1 - 2d\, e^{-\frac{m\epsilon}{2d}}.$$

Hence, for any $\delta > 0$,

$$P\left(\mathcal{R}(V_S) \leq \frac{2d}{m} \log \frac{2d}{\delta}\right) \geq 1 - \delta,$$

implying that $P(\mathcal{R}(V_S) \leq \epsilon) \geq 1 - \delta$ for $m \geq \frac{2d}{\epsilon} \log \frac{2d}{\delta}$.

The PAC Learning Framework

An alternative way to control sample complexity is to give a **generalization bound**.

A generalization bound states that, with probability at least $1-\delta$, $\mathcal{R}(R_S)$ is upper bounded by some quantity that depends on the sample size m and δ.

In the example of the axis-aligned rectangles, this is obtained by setting δ equal to the bound found above, that is,

$$\delta = 4\, e^{-m\frac{\epsilon}{4}} \quad \left(\delta = 2d\, e^{-m\frac{\epsilon}{2d}},\ \text{for the hypercube}\right)$$

and solving for ϵ.

This yields that, with probability at least $1-\delta$, the error of the algorithm is bounded by

$$\mathcal{R}(R_S) \leq \frac{4}{m}\log\frac{4}{\delta} \quad \left(\leq \frac{2d}{m}\log\frac{2d}{\delta},\ \text{for the hypercube}\right)$$

The PAC Learning Framework

Remark. The hypothesis set H we considered in the above example coincided with the concept class $\mathcal{C}$ and its cardinality was infinite. Nevertheless, the problem admitted a simple proof of PAC-learning.

This situation is not true in general because the specific geometric argument used in the proof cannot be extended in general.

Additionally, in the example of axis-aligned rectangles, the hypothesis h_S returned by the algorithm was always **consistent**.

Definition

A hypothesis h_S is **consistent with a set of training examples** S of a target concept c if and only if $h_S(x) = c(x)$ for each training example $x \in S$.

We present next a generalization bound, for consistent hypotheses, in the case where the cardinality $|H|$ of the hypothesis set is finite.

The PAC Learning Framework

Theorem: Learning Bound (finite H, consistent case)

Let H be a finite set of functions mapping from X to Y. Let $\mathcal{A}$ be an algorithm that, for any target concept $c \in H$ and for a i.i.d. sample S, returns a consistent hypothesis $h_S : \hat{\mathcal{R}}_S(h_S) = 0$. Then, for any $\epsilon, \delta > 0$, the inequality

$$\underset{S \sim \mathcal{D}^m}{P}(\mathcal{R}(h_S) \leq \epsilon) \geq 1 - \delta \quad \text{holds if} \quad m \geq \frac{1}{\epsilon}(\log|H| + \log\frac{1}{\delta}).$$

Equivalently, for any $\epsilon, \delta > 0$,

$$\underset{S \sim \mathcal{D}^m}{P}\left(\mathcal{R}(h_S) \leq \frac{1}{m}(\log|H| + \log\frac{1}{\delta})\right) \geq 1 - \delta$$

The sample complexity depends on $\frac{1}{\epsilon}, \frac{1}{\delta}$ and $|H|$.
The bound is independent of the algorithm $\mathcal{A}$, the target concept c or the distribution $\mathcal{D}$.

The PAC Learning Framework

Proof. We do not know which consistent hypothesis $h_S \in H$ is selectedby the algorithm. This hypothesis further depends on the training sample S.
Therefore, we need to give a uniform convergence bound, that is, a bound that holds for the set of all consistent hypotheses, which a fortiori includes h_S.

For any $\epsilon > 0$, define $H_\epsilon = \{h \in H : \mathcal{R}(h) > \epsilon\}$.

The probability that a hypothesis $h \in H$ is consistent on a training sample S drawn i.i.d., that is, that it would have no error on any point in S, can be bounded as

$$P(\hat{\mathcal{R}}_S(h) = 0) \leq (1 - \epsilon)^m$$

The PAC Learning Framework

Thus

$$\begin{aligned} P\left(\exists h \in H_\epsilon : \hat{\mathcal{R}}_S(h) = 0\right) &= P\left(\hat{\mathcal{R}}_S(h_1) = 0 \text{ or} \ldots \text{or } \hat{\mathcal{R}}_S(h_{|H_\epsilon|}) = 0\right) \\ &\leq \sum_{h \in H_\epsilon} P(\hat{\mathcal{R}}_S(h) = 0) \\ &\leq \sum_{h \in H_\epsilon} (1-\epsilon)^m \\ &\leq |H|(1-\epsilon)^m \\ &\leq |H| e^{-m\epsilon} \end{aligned}$$

The proof is completed by setting the RHS $= \delta$ and solving for ϵ.
$\square$

The PAC Learning Framework

Remarks. The theorem shows that, when the hypothesis set H is finite, a consistent algorithm $\mathcal{A}$ is a PAC-learning algorithm.

The generalization error of consistent hypotheses is upper bounded by a term that decreases as a function of the sample size m.

This is a general fact: *learning algorithms benefit from larger labeled training samples.*

The price to pay for coming up with a consistent algorithm is the use of a larger hypothesis set H containing target concepts. The upper bound in the theorem increases with $|H|$, albeit dependency is only logarithmic.

The PAC Learning Framework

Example: Universal concept class.
Let $X = \{0,1\}^n$ be the set of all Boolean n-component vectors and U_n be the concept class formed by all subsets of X.

- Is this concept class PAC-learnable?

To guarantee a consistent hypothesis, *the hypothesis class must include the concept class*, thus $|H| \geq |U_n| = 2^{2^n}$.
By the Learning Bound Theorem above,

$$m \geq \frac{1}{\epsilon}(\log|H| + \log\frac{1}{\delta}) \geq \frac{1}{\epsilon}(2^n \log 2 + \log\frac{1}{\delta})$$

Here, the number of training samples required is exponential in n, which is the cost of the representation of a point in X, hence PAC-learning is not guaranteed.
In fact, this universal concept class is not PAC-learnable.

The PAC Learning Framework

Let H be the hypothesis space and $h^* \in H$ be the optimal hypothesis

$$h^* = \underset{h \in H}{\operatorname{argmin}} \, \mathcal{R}(h)$$

Since we cannot evaluate the risk function $\mathcal{R}$ directly, we may instead approximate $\mathcal{R}$ using the empirical risk $\hat{\mathcal{R}}_S$ evaluated over the sample space S and approximate h^* by the hypothesis that minimizes the empirical risk

$$h_S = \underset{h \in H}{\operatorname{argmin}} \, \hat{\mathcal{R}}_S(h)$$

h_S may be suboptimal, but what is the gap?

$$\begin{aligned} \mathcal{R}(h_S) - \mathcal{R}(h^*) &= (\mathcal{R}(h_S) - \hat{\mathcal{R}}(h_S)) + (\hat{\mathcal{R}}(h_S) - \mathcal{R}(h^*)) \\ &\leq (\mathcal{R}(h_S) - \hat{\mathcal{R}}(h_S)) + (\hat{\mathcal{R}}(h^*) - \mathcal{R}(h^*)) \\ &= \sup_{h \in H} |\hat{\mathcal{R}}(h) - \mathcal{R}(h))| \end{aligned}$$

Hence we must control the difference $|\hat{\mathcal{R}}(h) - \mathcal{R}(h))|$.

The PAC Learning Framework

Case: finite hypothesis sets - inconsistent case.
In general, there may be no hypothesis in H consistent with the labeled training sample. This is the typical case in practice where the learning problems may be difficult or the concept classes more complex than the hypothesis set used by the learning algorithm.

In this case, Hoeffding's inequality in combination with the above observation that $\underset{S \sim \mathcal{D}^m}{E}[\hat{\mathcal{R}}_S(h)] = \mathcal{R}(h)$ gives that, for any fixed hypothesis $h : X \to \{0, 1\}$,

$$\underset{S \sim \mathcal{D}^m}{P} \left(|\hat{\mathcal{R}}_S(h) - \mathcal{R}(h)| \geq \epsilon \right) \leq 2\, e^{-2m\epsilon^2}$$

Setting RHS $= \delta$ and solving for ϵ gives the following bound.

Corollary (Generalization bound - fixed hypothesis)

Fix a hypothesis $h : X \to \{0, 1\}$. For any $\delta > 0$,

$$\underset{S \sim \mathcal{D}^m}{P} \left(|\mathcal{R}(h) - \hat{\mathcal{R}}_S(h)| \leq \sqrt{\frac{\log \frac{2}{\delta}}{2m}} \right) \geq 1 - \delta.$$

The PAC Learning Framework

Example. Imagine tossing a biased coin that lands heads with probability p, and let our hypothesis be the one that always guesses tails.
Then the true error rate is $\mathcal{R}(h) = p$ and the empirical error rate $\hat{\mathcal{R}}_S(h) = \hat{p}$, where $\hat{p}$ is the empirical probability of heads based on the training sample drawn i.i.d.
According to the generalization bound, for any $\delta > 0$, with probability at least $1 - \delta$,

$$|p - \hat{p}| \leq \sqrt{\frac{\log \frac{2}{\delta}}{2m}}$$

If we set $\delta = 0.02$ and choose a sample of size $m = 500$, with probability at most 98%, the following bound is guaranteed

$$|p - \hat{p}| \leq \sqrt{\frac{\log 10}{1000}} \approx 0.048.$$

Remark. We cannot use this estimate to bound the generalization error of the hypothesis h_S returned by a learning algorithm when training on a sample S *since h_S is not a fixed hypothesis but a random variable* depending on the training sample S drawn.

The PAC Learning Framework

Unlike the case of a fixed hypothesis for which the expectation of the empirical error is the generalization error (we used the observation that, for h fixed, $\underset{S\sim\mathcal{D}^m}{E}[\hat{\mathcal{R}}_S(h)] = \mathcal{R}(h)$), now the generalization error $\mathcal{R}(h_S)$ is a random variable and, in general, distinct from the expectation $E[\mathcal{R}_S(h_S)]$, which is a constant.

Theorem (Learning Bound - finite H, inconsistent case)

Let H be a finite hypothesis set. Then, for any $\delta > 0$, with probability at least $1 - \delta$, the following inequality holds for all $h \in H$

$$\mathcal{R}(h) \leq \hat{\mathcal{R}}_S(h) + \sqrt{\frac{\log|H| + \log\frac{2}{\delta}}{2m}}$$

The PAC Learning Framework

Proof. Let $h_1, \dots, h_{|H|}$ be the elements of H. Using the observation valid for a single hypothesis, we obtain

$$\begin{aligned}
& P\left(\exists h \in H : |\hat{\mathcal{R}}_S(h) - \mathcal{R}_S(h)| > \epsilon\right) \\
&= P\left(|\hat{\mathcal{R}}_S(h_1) - \mathcal{R}_S(h_1)| > \epsilon \text{ or} \dots \text{or } |\hat{\mathcal{R}}_S(h_{|H_\epsilon|}) - \mathcal{R}_S(h_{|H_\epsilon|})| > \epsilon\right) \\
&\leq \sum_{h \in H} P(|\hat{\mathcal{R}}_S(h) - \mathcal{R}_S(h)| > \epsilon) \\
&\leq 2|H| e^{-2m\epsilon^2}
\end{aligned}$$

The proof follows by setting the RHS $= \delta$ and solving for ϵ. $\square$

The PAC Learning Framework

Remark. The theorem shows that, for a finite hypothesis set H,

$$\mathcal{R}(h) \leq \hat{\mathcal{R}}_S(h) + O\left(\sqrt{\frac{\log |H|}{m}}\right)$$

where $\log |H|$ can be interpreted as the number of bits needed to represent H.

As in the consistent case, a larger sample size m guarantees better generalization even though here the bound is a less favorable function of $\frac{\log |H|}{m}$ as it varies as the square root of this term.

The bound suggests seeking a trade-off between reducing the empirical error versus controlling the size of the hypothesis set: a larger hypothesis set is penalized by the second term but could help reduce the empirical error, that is the first term. But, for a similar empirical error, it suggests using a smaller hypothesis set.

Rademacher Complexity

Unfortunately, the sample complexity bounds discussed above are uninformative when dealing with **infinite hypothesis sets** - which is the typical situation found in machine learning applications.

Problem: *How can we achieve efficient learning from a finite sample if the hypothesis set H is infinite?*

The general idea consists essentially of reducing the infinite case to the analysis of finite sets of hypotheses.

There are different techniques for that reduction, each relying on a different notion of complexity for the family of hypotheses.
One such technique is **Rademacher complexity**.

I will show that the computation of the empirical Rademacher complexity is impractical (NP-hard) for some hypothesis sets.
Thus, I will subsequently introduce two other notions, the **growth function** and the **VC-dimension**.

Rademacher Complexity

Definition (Rademacher complexity)

Given a set of vectors $A \subset \mathbb{R}^m$ the **Rademacher complexity** is defined as

$$\mathfrak{R}_m(A) = \underset{\sigma}{E}\left[\sup_{a \in A} \frac{1}{m} \sum_{i=1}^{m} \sigma_i a_i\right]$$

where the expectation is taken over $\sigma = (\sigma_1, \ldots, \sigma_m)^t$ with σ_i independent random variables taking values in ± 1 according to the Rademacher distribution, i.e., $P(\sigma_i = 1) = P(\sigma_i = -1) = \frac{1}{2}$

Examples:
Let $A = \{(a_1, a_2)\} \subset \mathbb{R}^2$ (a single vector). Then
$\mathfrak{R}_1(A) = \frac{1}{2}(\frac{1}{4}(a_1 + 1_2) + \frac{1}{4}(a_1 - 1_2) + \frac{1}{4}(-a_1 + 1_2) + \frac{1}{4}(-a_1 - 1_2)) = 0$

Let $A = \{(1,1),(1,2)\} \subset \mathbb{R}^2$. Then

$$\begin{aligned}\mathfrak{R}_2(A) &= \frac{1}{2} \cdot \frac{1}{4}(\max(1+1, 1+2) + \max(1-1, 1-2) \\ &+ \max(-1+1, -1+2) + \max(-1-1, -1-2)) = \frac{1}{4}\end{aligned}$$

Rademacher Complexity

Notation:

X : input space

Y : target space

H hypothesis space

$\mathcal{G} = \{g : (x, y) \mapsto L(h(x), y), h \in H\}$: family of loss functions associated to H

Definition (Rademacher complexity of function class)

Let $\mathcal{G}$ be a family of (measurable) loss functions associated to H and $S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \subset X \times Y$ be a fixed sample of size m. The **empirical Rademacher complexity** of $\mathcal{G}$ with respect to the sample S is defined as

$$\mathfrak{R}_S(\mathcal{G}) = \underset{\sigma}{E}\left[\sup_{g \in \mathcal{G}} \frac{1}{m} \sum_{i=1}^{m} \sigma_i g(x_i, y_i)\right]$$

where $\sigma = (\sigma_1, \ldots, \sigma_m)^t$ with σ_i being Rademacher random variables.

Rademacher Complexity

Interpretation:
Let $g_S = (g(x_1, y_1), \ldots, g(x_m, y_m))^t$
The **empirical Rademacher complexity** of $\mathcal{G}$ with respect to the sample S can be written as

$$\mathfrak{R}_S(\mathcal{G}) = \underset{\sigma}{E}\left[\sup_{g \in \mathcal{G}} \frac{\sigma \cdot g_S}{m}\right]$$

The inner product $\sigma \cdot g_S$ measures the correlation of g_S with the vector of random noise σ (Rademacher r.v.).

Hence $\mathfrak{R}_S(\mathcal{G})$ *measures on average how well the function class $\mathcal{G}$ correlates with random noise on S.*
Richer or more complex families $\mathcal{G}$ can generate more vectors g_S and thus better correlate with random noise, on average.

Rademacher Complexity

Definition (Rademacher complexity)

Let $\mathcal{D}$ be the distribution according to which samples are drawn. For any $m \geq 1$, the **Rademacher complexity** of $\mathcal{G}$ is the expectation of the empirical Rademacher complexity over all samples of size m drawn according to $\mathcal{D}$:

$$\mathfrak{R}_m(\mathcal{G}) = \underset{S \sim \mathcal{D}^m}{E} [\mathfrak{R}_S(\mathcal{G})]$$

The Rademacher complexity measures the richness of the class of loss functions $\mathcal{G}$. We can motivate the Rademacher complexity from the binary classification. Let $\mathcal{G}$ be the classification functions mapping the data to its label $\sigma_i \in \{-1, 1\}$. One can show that $\sup_{g \in \mathcal{G}} \sum_{i=1}^{m} \sigma_i g(x_i, y_i)$ is equivalent to minimizing the classification error. Taking the expectation over all σ amounts to considering all possible labeling of the samples. If $\mathcal{G}$ consists of a single function, then $\mathfrak{R}_m(\mathcal{G}) = 0$; if $\mathcal{G}$ shatters any sample of size m, then $\mathfrak{R}_m(\mathcal{G}) = 1$.

Rademacher Complexity

We have the following generalization bounds based on Rademacher complexity.

Theorem

Let $\mathcal{G}$ be a family of functions mapping from $X \times Y \to [0,1]$. Then, for any $\delta > 0$, with probability at least $1-\delta$ over the draw of i.i.d. samples of size m, for all $g \in \mathcal{G}$ we have

$$E[g(x,y)] \leq \frac{1}{m}\sum_{i=1}^{m} g(x_i,y_i) + 2\,\mathfrak{R}_m(\mathcal{G}) + \sqrt{\frac{\log\frac{1}{\delta}}{2m}}$$

$$E[g(x,y)] \leq \frac{1}{m}\sum_{i=1}^{m} g(x_i,y_i) + 2\,\mathfrak{R}_m(\mathcal{G}) + 3\sqrt{\frac{\log\frac{2}{\delta}}{2m}}$$

Compared with the generalization bound stated in the section above, the term $\sqrt{\frac{\log|H|}{m}}$ is replaced by the Rademacher complexity $\mathfrak{R}_m(\mathcal{G})$.

Rademacher Complexity

The proof of the theorem requires the following result from probability (a refinement of Hoeffding's inequality).

McDiarmid's inequality

Let $x_1, \ldots, x_m \in X$ be a set of $m \geq 1$ independent random variables and assume that there exist constants $c_1, \ldots, c_m$ such that $f : X^m \mapsto \mathbb{R}$ satisfies

$$|f(x_1, \ldots, x_i, \ldots, x_m) - f(x_1, \ldots, x_i', \ldots, x_m)| \leq c_i$$

for all i and any points $x_1, \ldots, x_m, x_i' \in X$. Then, with the notation $f(S) = f(x_1, \ldots, x_m)$, for any $\epsilon > 0$, we have

$$P\left(f(S) - E[f(S)] \geq \epsilon\right) \leq \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^{m} c_i^2}\right)$$

$$P\left(f(S) - E[f(S)] \leq -\epsilon\right) \leq \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^{m} c_i^2}\right)$$

Rademacher Complexity

A straightforward calculation yields the following corollary.

Corollary (McDiarmid's inequality)

Under the assumptions above, suppose, in addition, that $c_i \leq \frac{1}{m}$ for all i. Then

$$P\left(|f(S) - E[f(S)]| \geq \epsilon\right) \leq 2e^{-2\epsilon^2 m}.$$

Remark. By setting the RHS $= \delta$ and solving for ϵ, it follows that, for any $\delta > 0$,

$$P\left(|f(S) - E[f(S)]| \leq \sqrt{\frac{\log \frac{2}{\delta}}{2m}}\right) \geq 1 - \delta$$

Rademacher Complexity

Proof of Theorem. For a sample $S = ((x_1, y_1), \ldots, (x_m, y_m))$ and any $g \in \mathcal{G}$, we denote the empirical average of g over S by

$$\hat{E}_s[g] = \frac{1}{m} \sum_{i=1}^{m} g(x_i, y_i).$$

The main idea of the proof consists of applying McDiarmid's inequality to the function Φ defined, for any sample S, as

$$\Phi(S) = \sup_{g \in \mathcal{G}} E[g] - \hat{E}_S[g]$$

Rademacher Complexity

Let S and S' be two samples differing by exactly one point, say (x_i, y_i).
Since the difference of suprema does not exceed the supremum of the difference, we have

$$|\Phi(S') - \Phi(S)| \leq \sup_{g \in \mathcal{G}} |\hat{E}_S[g] - \hat{E}_{S'}[g]| \leq \frac{1}{m} \sup_{g \in \mathcal{G}} |g(x_i, y_i) - g(x_i', y_i')| \leq \frac{1}{m}$$

By McDiarmid's inequality (Corollary), we have that, for any $\delta > 0$, with probability at least $1 - \frac{\delta}{2}$,

$$\Phi(S) \leq \underset{S}{E}[\Phi(S)] + \sqrt{\frac{\log \frac{2}{\delta}}{2m}}$$

Rademacher Complexity

Next, observing that points in S' are sampled i.i.d. and, thus, $E[g] = E_{S'}[\hat{E}_{S'}(g)]$, we have

$$\begin{aligned}
\underset{S}{E}[\Phi(S)] &= \underset{S}{E}[\sup_{g\in\mathcal{G}}(E[g] - \hat{E}_S(g))] \\
&= \underset{S}{E}[\sup_{g\in\mathcal{G}} \underset{S'}{E}[\hat{E}_{S'}(g) - \hat{E}_S(g)]] \\
&\le \underset{S,S'}{E}[\sup_{g\in\mathcal{G}}(\hat{E}_{S'}(g) - \hat{E}_S(g)) \\
&= \underset{S,S'}{E}[\sup_{g\in\mathcal{G}} \frac{1}{m}\sum_{i=1}^{m}(g(x_i', y_i') - g(x_i, y_i))] \\
&= \underset{\sigma,S,S'}{E}[\sup_{g\in\mathcal{G}} \frac{1}{m}\sum_{i=1}^{m}\sigma_i(g(x_i', y_i') - g(x_i, y_i))] \\
&= \underset{\sigma,S'}{E}[\sup_{g\in\mathcal{G}} \frac{1}{m}\sum_{i=1}^{m}\sigma_i(g(x_i', y_i')]| + \underset{\sigma,S}{E}[\sup_{g\in\mathcal{G}} \frac{1}{m}\sum_{i=1}^{m}(-\sigma_i)(g(x_i, y_i)] \\
&= 2\underset{\sigma,S}{E}[\sup_{g\in\mathcal{G}} \frac{1}{m}\sum_{i=1}^{m}\sigma_i(g(x_i, y_i)]| = 2\,\mathfrak{R}_m(\mathcal{G})
\end{aligned}$$

Rademacher Complexity

To derive a bound in terms of $\mathfrak{R}_S(\mathcal{G})$, we observe that, by definition, changing one point in S changes $\mathfrak{R}_S(\mathcal{G})$ by at most $\frac{1}{m}$.

By applying Mc Diarmid's inequality again, we have that, for any $\delta > 0$, with probability $1 - \delta/2$

$$\mathfrak{R}_m(\mathcal{G}) \leq \mathfrak{R}_S(\mathcal{G}) + \sqrt{\frac{\log \frac{2}{\delta}}{2m}}$$

By using the union bounds and combining the estimate above with the estimate for $\Phi(S)$ we obtain that, for any $\delta > 0$, with probability at least $1 - \delta$, we have

$$\Phi(S) \leq \mathfrak{R}_m(\mathcal{G}) + 3\sqrt{\frac{\log \frac{2}{\delta}}{2m}} \qquad \square$$

Rademacher Complexity

Remark. The key idea of the proof is in the step
$\underset{S,S'}{E}[\sup_{g\in\mathcal{G}} \frac{1}{m}\sum_{i=1}^{m}(g(x_i', y_i') - g(x_i, y_i))] = \underset{\sigma,S,S'}{E}[\sup_{g\in\mathcal{G}} \frac{1}{m}\sum_{i=1}^{m}\sigma_i(g(x_i', y_i') - g(x_i, y_i))]$
where we introduce the Rademacher random variables.
Setting $\sigma_i = -1$ has the same effect as swapping x_i and x_i'. To see why this step is allowed, imagine that after choosing S and S' we update both sets by swapping x_i and x_i' with probability $1/2$ for $i = 1, \ldots, m$. Now this swapping operation leaves the distribution over pairs of sets S and S' unaffected: each possible outcome of S and S' has the same probability before and after the swapping procedure. Thus the expected values on the two sides are equal.

Rademacher Complexity

The following result relates the empirical Rademacher complexities to the family of loss functions associated the case of binary loss.

Proposition

Let H be a family of (measurable) functions taking values in $\{-1, +1\}$ and let $\mathcal{G}$ be the family of loss functions associated to H for the zero-one loss:

$$\mathcal{G} = \{(x, y) \mapsto 1_{h(x) \neq y}, h \in H\}.$$

For any sample $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ of elements in $X \times \{-1, +1\}$, let S_X denote its projection over X, that is $S_X = (x_1, \ldots, x_m)$. Then, the following relation holds

$$\mathfrak{R}_S(\mathcal{G}) = \frac{1}{2} \mathfrak{R}_{S_X}(H).$$

Rademacher Complexity

Proof of Proposition.
For $S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \in X \times \{-1, +1\}$ the empirical Rademacher complexity of $\mathcal{G}$ can be written as

$$\begin{aligned}
\mathfrak{R}_S(\mathcal{G}) &= \underset{\sigma}{E}[\sup_{h \in H} \tfrac{1}{m} \sum_{i=1}^{m} \sigma_i \, 1_{h(x_i) \neq y_i}] \\
&= \underset{\sigma}{E}[\sup_{h \in H} \tfrac{1}{m} \sum_{i=1}^{m} \sigma_i \, \tfrac{1 - y_i \, h(x_i)}{2}] \\
&= \tfrac{1}{2} \underset{\sigma}{E}[\sup_{h \in H} \tfrac{1}{m} \sum_{i=1}^{m} (-\sigma_i) \, y_i \, h(x_i)] \\
&= \tfrac{1}{2} \underset{\sigma}{E}[\sup_{h \in H} \tfrac{1}{m} \sum_{i=1}^{m} \sigma_i \, h(x_i)] = \tfrac{1}{2} \mathfrak{R}_{S_X}(H),
\end{aligned}$$

where we used the fact that, for a fixed $y_i \in \{-1, +1\}$, σ_i and $(-\sigma_i \, y_i)$ are distributed in the same way. $\square$

Rademacher Complexity

By taking expectations, the last proposition implies that, for any $m \geq 1$, $\mathfrak{R}_m(\mathcal{G}) = \frac{1}{2}\mathfrak{R}_m(H)$. Hence, combining the proposition and the theorem above, we derive the following bounds.

Theorem - Rademacher complexity bounds for binary classification

Let H be a family of (measurable) functions taking values in $\{-1, +1\}$ and let $\mathcal{D}$ be be the distribution over the input space X. Then, for any $\delta > 0$, with probability at least $1 - \delta$ over a sample S of size m drawn according to $\mathcal{D}$, for any $h \in H$ we have

$$\mathcal{R}(h) \leq \hat{\mathcal{R}}_S(h) + \mathfrak{R}_m(H) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}}$$

$$\mathcal{R}(h) \leq \hat{\mathcal{R}}_S(h) + \mathfrak{R}_m(H) + 3\sqrt{\frac{\log \frac{2}{\delta}}{2m}}$$

Growth function

The growth function measures the richness of a hypothesis class.

Definition

The **growth function** or **shatter coefficient** $Gr_H : \mathbb{N} \to \mathbb{N}$ for a hypothesis set H is defined by

$$Gr_H(m) = \max_{\{x_1,\dots,x_m\}\in X} |\{h(x_1), \dots, h(x_m) \colon h \in H\}|,$$

and it counts the maximum number of distinct ways in which m points can be classified using the hypotheses h in H.

Unlike the Rademacher complexity, the growth function does not depend on the distribution but it is purely combinatorial.

Growth function

Let $S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \subset X \times Y$ and H be the hypothesis class.

Definition

Given a set $x_1, \ldots, x_m \subset X$, the **dichotomy** generated by $h \in H$ is the sequence

$$(h(x_1), \ldots, h(x_m)).$$

Thus, Gr_H counts the maximum number of dichotomies realized by the hypothesis space H on any set of m points from the input space X.

Theorem

Let H be the hypothesis class. For any set of size m, we have

$$Gr_H(m) \leq 2^m.$$

Growth function

Example. Consider the hypothesis class H consisting of one-dimensional threshold functions and let $S = \{1, 2, 3, 4, 5, 6\}$.

It is easy to see that have we have 7 dichotomies realized by this hypothesis class.

Lemma (Growth function for one-dimensional threshold function)

Let H be the hypothesis class of one-dimensional threshold functions and let $S = \{x_1, \dots, x_m\}$ be a list of numbers. Then

$$Gr_H(m) = m + 1.$$

Growth function

Theorem (Massart's Lemma)

Let $\mathcal{A} \subset \mathbb{R}^m$ be a finite set, with $r = \max_{x \in \mathcal{A}} \|x\|_2$. Then

$$\underset{\sigma}{E}[\tfrac{1}{m} \sup_{x \in \mathcal{A}} \sum_{i=1}^{m} \sigma_i \, x_i] \leq \frac{r\sqrt{2 \log |\mathcal{A}|}}{m},$$

where the terms σ_i are independent uniform random variables taking values in $\{-1, +1\}$ and $x = (x_1, \ldots, x_m)$.

Growth function

Massart's Lemma follows from the Maximal inequality below observing that the random variables $\sigma_i x_i$ are independent and take values in $[-|x_i|, +|x_i|]$ with $\sqrt{\sum_{i=1}^m x_i^2} \leq r$.

Maximal inequality

Let $X_1, \ldots, X_n$ be real-valued random variables such that, for all $j \in [n]$, $X_j = \sum_{i=1}^m Y_{i,j}$ where, for each fixed $j \in [n]$, the terms $Y_{i,j}$ are independent zero mean random variables taking values in the interval $[-r_i, +r_i]$ for some $r_i > 0$. Then

$$E[\max_{j \in [n]} X_j] \leq r\sqrt{2 \log n},$$

with $r = \sqrt{\sum_{i=1}^m r_i^2}$.

Growth function

A corollary of Massart's Lemma is that we can bound the Rademacher complexity using the growth function.

Theorem

Let $\mathcal{G}$ be a family of (measurable) loss functions taking values in $\{-1, +1\}$, then

$$\mathfrak{R}_m(\mathcal{G}) \le \sqrt{\frac{2 \log Gr_{\mathcal{G}}(m)}{m}}.$$

Growth function

Proof. For a fixed sample $S = (x_1, \dots, x_m)$, we denote by $\mathcal{G}_S$ the set of vectors of function values $(g(x_1), \dots, g(x_m))$ where $g \in \mathcal{G}$.

Since each g takes values in $\{-1, +1\}$, the norm of these vectors is bounded by $\sqrt{m}$. By Massart's Lemma, it follows that

$$\mathfrak{R}_m(\mathcal{G}) = \underset{S}{E}\left[\underset{\sigma}{E}\left[\sup_{g \in \mathcal{G}_S} \frac{1}{m} \sum_{i=1}^{m} \sigma_i g(x_i)\right]\right] \leq \underset{S}{E}\left[\frac{\sqrt{m}\sqrt{2}\log|\mathcal{G}_S|}{m}\right]$$

By definition, $|\mathcal{G}_S|$ is bounded by the growth function. Hence

$$\mathfrak{R}_m(\mathcal{G}) \leq \underset{S}{E}\left[\frac{\sqrt{m}\sqrt{2\log Gr_{\mathcal{G}}(m)}}{m}\right] = \sqrt{\frac{2\log Gr_{\mathcal{G}}(m)}{m}} \qquad \square$$

Growth function

Combining the Rademacher complexity bound with the last observation yields immediately the following generalization bound in terms of the growth function.

Corollary (Growth function generalization bound)

Let H be a family of (measurable) loss functions taking values in $\{-1, +1\}$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, for any $h \in H$ we have

$$\mathcal{R}(h) \leq \hat{\mathcal{R}}_S(h) + \sqrt{\frac{2 \log Gr_H(m)}{m}} + \sqrt{\frac{\log \frac{1}{\delta}}{2m}}$$

VC dimension

The computation of the growth function is not always practical since, by definition, it requires computing $Gr(m)$ for all $m \geq 1$.

The **VC-dimension (Vapnik-Chervonenkis dimension)** is also a purely combinatorial notion that is directly related to the growth function but it is often easier to compute.

Recall that, given a hypothesis set H, a **dichotomy** of a set S is one of the possible ways of labeling all points of S using a hypothesis $h \in H$.

Given a set S of $m \geq 1$ points, we say that H **shatters** S when H realizes all possible dichotomies of S, that is when $Gr(m) = 2^m$.

VC dimension

Definition

The **VC dimension** of a hypothesis set H is the cardinality of the largest set S that can be shattered by H, that is

$$VC\dim(H) = max\{m : Gr_H(m) = 2^m\}.$$

If arbitrarily large finite sets can be shattered by H, then $VC\dim(H) = \infty$.

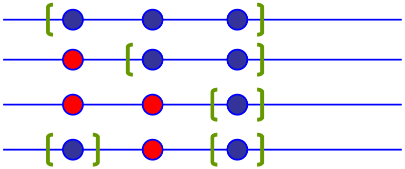
Note: The definition of VC dimension states that, if there exists a set of d points that can be shattered by H and there is no set of $d+1$ points that can be shattered by H, then $VC\dim(H) = d$. The definition does not require that any set of d points can be shattered by H.

VC dimension

Example: Interval hypothesis class. Let H be the set of intervals on the real line such that $h(x) = 1$ iff x is in the interval.

It is easy to see that H shatters a set consisting of 2 points.

However, it cannot shatter 3 points



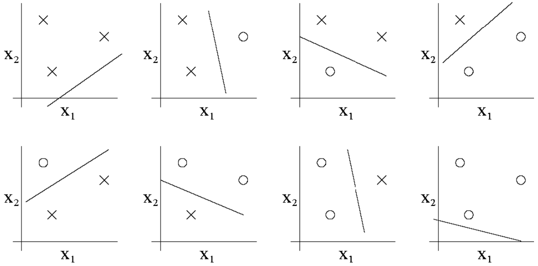
Thus $VC\dim(H) = 2$

VC dimension

Example: Linear classifiers in $\mathbb{R}^2$. Let H be the set of linear classifiers in $\mathbb{R}^2$. That is, we want to learn $c : X \mapsto \{-1, +1\}$ using the hypothesis set of lines in $\mathbb{R}^2$.

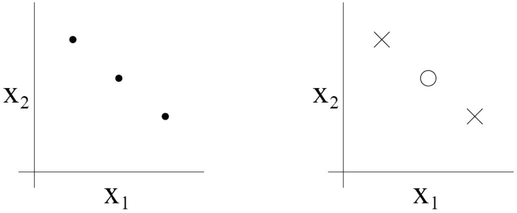
What is the VC dimension of H?

A set of 3 non-collinear points in $\mathbb{R}^2$ can be shattered by H, that is, we can realize all possible $2^3 = 8$ dichotomies over these points.



VC dimension

Note that 3 collinear points in $\mathbb{R}^2$ cannot be shattered by H as shown below



This does not exclude that $VC\text{dim}(H) = 3$.

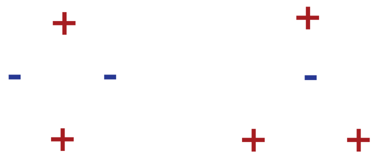
Note. As we remarked, $VC\text{dim}(H) = d$ does not imply that all sets of size d or less are shattered and, in fact, this is typically not the case. It is sufficient that one set of size d is shattered by H.

VC dimension

To conclude that $VC\dim(H) = 3$, we need to show that four points in $\mathbb{R}^2$ cannot be shattered. The argument is give below. After removing the case with 3 collinear points, we are left with two cases:

(i) The four points lie on their convex hull, In this case, a positive labeling for one diagonal pair and a negative labeling for the other diagonal pair cannot be realized.

(ii) Three of the four points lie on the convex hull and the remaining point is internal. In this case, a labeling which is positive for the points on the convex hull and negative for the interior point cannot be realized.



VC dimension

We can extend the result to $\mathbb{R}^d$ to show that $VC\dim(H) = d + 1$.

Example: Hyperplane classifiers in $\mathbb{R}^d$. Let H be the set of linear classifiers in $\mathbb{R}^d$. That is, we want to learn $c : X \mapsto \{-1, +1\}$ using the hypothesis consisting of hyperplanes in $\mathbb{R}^d$.

We derive a lower bound of $VC\dim(H)$ by starting with a set of $d + 1$ points in $\mathbb{R}^d$, setting x_0 to be the origin and defining $x_i \in \mathbb{R}^d$, for $i \in \{1, 2, \ldots, d\}$ as the point whose i-th coordinate is 1 and all others are 0.
Let $y_0, y_1, \ldots, y_d \in \{-1, +1\}$ be an set of labels for $x_0, x_1, \ldots, x_d$. Let w be the vector whose i-th coordinate is y_i. Then the classifier defined by the hyperplane of equation

$$w \cdot x + \frac{y_0}{2}$$

shatters $x_0, x_1, \ldots, x_d$ since, for each $i \in \{0, \ldots, d\}$

$$\text{sgn}\left(w \cdot x_i + \frac{y_0}{2}\right) = \text{sgn}(y_i + \frac{y_0}{2}) = y_i$$

VC dimension

To obtain an upper bound, we show that no set of $d+2$ points can be shattered by half space using Radon's Theorem.

Theorem (Radon's theorem)

Any set X of $d+2$ points in $\mathbb{R}^d$ can be partitioned into two subsets X_1 and X_2 such that the convex hulls of X_1 and X_2 intersect.

By Radon's theorem, a set X of $d+2$ points can be partitioned into two sets X_1 and X_2 such that their convex hulls intersect.
When two sets of points X_1 and X_2 are separated by a hyperplane, their convex hulls are also separated by that hyperplane.
Thus, X_1 and X_2 cannot be separated by a hyperplane and X is not shattered.
Combining our lower and upper bounds, we have proven that $VC\text{dim}(H) = d+1$ when H is the set of hyperplanes in $\mathbb{R}^d$. $\square$

VC dimension

Proof of Radon's theorem. Let $X = \{x_1, \ldots, x_{d+2}\}$ and consider the system of equations

$$\sum_{i=1}^{d+2} \alpha_i x_i = 0 \quad \text{and} \quad \sum_{i=1}^{d+2} \alpha_i = 0 \qquad (1)$$

This is a system of $d+1$ equation (d equations from the first, one for each component, 1 from the second one) and $d+2$ unknowns. Hence the system admits a non-zero solution $\beta_1, \ldots, \beta_{d+1}$.

Since $\sum_{i=1}^{d+2} \beta_i = 0$, both sets $J_1 = \{i \in [d+1] : \beta_i > 0\}$ and $J_2 = \{i \in [d+1] : \beta_i \leq 0\}$ are non-empty.

In addition, the sets

$$X_1 = \{x_1 : i \in J_1\} \text{ and } X_2 = \{x_1 : i \in J_2\}$$

form a partition of X.

VC dimension

Let $\beta = \sum_{i \in J_1} \beta_i$.
Since $\sum_{i=1}^{d+2} \beta_i = 0$, then $\beta = \sum_{i \in J_1} \beta_i = -\sum_{i \in J_2} \beta_i$.
Thus, by the first equation in (1)

$$\sum_{i \in J_1} \frac{\beta_i}{\beta} x_i = \sum_{i \in J_2} \frac{-\beta_i}{\beta} x_i$$

with $\sum_{i \in J_1} \frac{\beta_i}{\beta} = \sum_{i \in J_2} \frac{-\beta_i}{\beta} = 1$ and $\beta_i/\beta \geq 0$ for $i \in J_1$ and $-\beta_i/\beta \geq 0$ for $i \in J_2$.

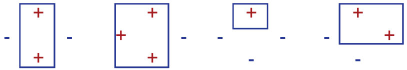
By the definition of convex hull, this implies that $\sum_{i \in J_1} \frac{\beta_i}{\beta} x_i$ belongs to both the convex hull of X_1 and X_2. $\square$

Definition. For $X \subset \mathbb{R}^n$, the **convex hull** of X is

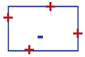
$$conv(X) = \left\{ \sum_{i=1}^{m} \alpha_i x_i : m \geq 1, x_i \in X, \alpha_i \geq 0, \sum_{1}^{m} \alpha_i = 1 \right\}$$

VC dimension

Example: Axis-aligned Rectangles. We show first that $VC\text{dim}(H) \geq 4$ by considering 4 points in a diamond pattern. As the figure shows (4 representative cases) all $2^4 = 16$ dichotomies can be realized.



For any set of five distinct points, if we construct the minimal axis-aligned rectangle containing these points, one of the five points is in the interior of the rectangle.

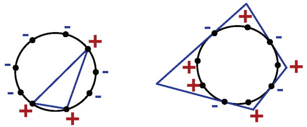


If we assign a negative label to this interior point and a positive label to each of the remaining four points, there is no axis-aligned rectangle that can realize this labeling. Hence, no set of five distinct points can be shattered. Thus $VC\text{dim}(H) = 4$.

VC dimension

Example: Convex d-gons in $\mathbb{R}^2$. The hypothesis class consists of convex polygons with d sides. We will show that it shatters $2d+1$ points. [Note: rectangles above were required to be aligned]

To get a lower bound, we select $2d+1$ points that lie on a circle, and for a particular labeling, if there are more negative than positive labels, then the points with the positive labels are used as the polygon's vertices; otherwise, the tangents of the negative points serve as the edges of the polygon.



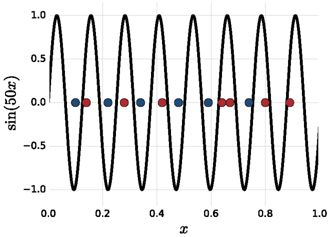
|positive points| < |negative points| |positive points| > |negative points|

To derive an upper bound, it can be shown that choosing points on the circle maximizes the number of possible dichotomies, and thus $VC\dim(H) = 2d+1$.

VC dimension

Remark. The examples presented above appear to suggest that the VC dimension of H coincides with the number of free parameters defining H. For example, the number of parameters defining hyperplanes matches their VC dimension.
However, this situation does not hold in general.

Example. Sine functions. Consider the hypothesis class H of function $f_\alpha(x) = \sin(\alpha x)$, $\alpha \in \mathbb{R}$.
We use f_α to classify the points on the real line: a point is labeled positively if it is above the curve, negatively otherwise.



VC dimension

Although this family of functions f_α is defined via a single parameter, it can be shown that $VC\dim(H) = \infty$.

Given any $m \in \mathbb{N}$, let $x_i = 10^{-i}$, $i = 1, \ldots, m$

Let $y_1, \ldots, y_m$ be a set of labels in $\{-1, +1\}$.

Then $\operatorname{sgn}(f_\alpha(x))$ gives the correct labels if we choose α to be

$$\alpha = \pi(1 + \sum_{i=1}^{m} \frac{(1 - y_i)10^i}{2})$$

Since m is arbitrary, this shows that $VC\dim(H) = \infty$.

VC dimension

How is the VC dimension related to the cardinality of the hypothesis class?

Proposition

For every finite hypothesis classes H, we have $VC\dim(H) < \log|H|$.

Proof. Suppose that $VC\dim(H) = d$. This implies that $Gr_H(d) = 2^d$.

Also, for every set of size $m > 1$, we have that $Gr_H(m) \leq |H|$. Hence, it must be $2^d = Gr_H(d) \leq |H|$.

By taking log on both sides, we conclude that

$$d \leq \log|H|.$$

VC dimension

The next result clarifies the connection between the notions of growth function and VC dimension.

Theorem (Sauer's lemma)

Let H be a hypothesis set with $VC\dim(H) = d$. Then, for all $m \in \mathbb{N}$ the following inequality holds:

$$Gr_H(m) \leq \sum_{i=0}^{d} \binom{m}{i}$$

Sauer's lemma can be proved using an argument by induction.

VC dimension

The significance of Sauer's lemma can be seen below.

Corollary

Let H be a hypothesis set with $VC\text{dim}(H) = d$. Then for all $m \geq d$

$$Gr_H(m) \leq \left(\frac{em}{d}\right)^d = O(m^d)$$

This shows that the growth function only exhibits two types of behavior:

- either $VC\text{dim}(H) = d < \infty$, in which case $Gr_H(m) = O(m^d)$;
- or $VC\text{dim}(H) = \infty$, in which case $Gr_H(m) = 2^m$.

VC dimension

Proof. Using Sauer's lemma we have

$$\begin{aligned}
Gr_H(m) &\leq \sum_{i=0}^{d} \binom{m}{i} \\
&\leq \sum_{i=0}^{d} \binom{m}{i} (\tfrac{m}{d})^{d-i} \\
&= (\tfrac{m}{d})^{d} \sum_{i=0}^{d} \binom{m}{i} (\tfrac{d}{m})^{i} \\
&= (\tfrac{m}{d})^{d} (1 + \tfrac{d}{m})^{m} \quad \text{(binomial thm.)} \\
&\leq (\tfrac{m}{d})^{d} (e^{d/m})^{m} \quad \text{(ineq: } (1 = +x) \leq e^{x}) \\
&= (\tfrac{m}{d})^{d} e^{d} = (\tfrac{me}{d})^{d} \qquad \square
\end{aligned}$$

VC dimension

Using the relationship between VC dimension and the growth function, we derive the following generalization bounds based on the VC dimension.

Corollary (VC dimension generalization bounds)

Let H be a family of functions taking values in $\{-1, +1\}$ with VC dimension d. Then, for any $\delta > 0$, with probability at least $1 - \delta$, the following holds for all $h \in H$

$$\mathcal{R}(h) \leq \hat{\mathcal{R}}_S(h) + \sqrt{\frac{2d \log \frac{m}{d}}{m}} + \sqrt{\frac{\log \frac{1}{\delta}}{2m}}$$

This shows that the generalization bound is of the form

$$\mathcal{R}(h) \leq \hat{\mathcal{R}}_S(h) + O\left(\sqrt{\frac{\log \frac{m}{d}}{\frac{m}{d}}}\right) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}}$$

emphasizing the importance of the ratio $\frac{m}{d}$ for generalization.

VC dimension

Proof. Using the growth function generalization bound, we have that

$$\mathcal{R}(h) \leq \hat{\mathcal{R}}_S(h) + \sqrt{\frac{2\log Gr_H(m)}{m}} + \sqrt{\frac{\log\frac{1}{\delta}}{2m}}$$

Using Sauer's lemma, it follows that

$$\begin{aligned}\mathcal{R}(h) &\leq \hat{\mathcal{R}}_S(h) + \sqrt{\frac{2\log(\frac{me}{d})^d}{m}} + \sqrt{\frac{\log\frac{1}{\delta}}{2m}} \\ &= \hat{\mathcal{R}}_S(h) + \sqrt{\frac{2d\log\frac{me}{d}}{m}} + \sqrt{\frac{\log\frac{1}{\delta}}{2m}}.\end{aligned}$$

□

VC dimension

Lower bound estimates are shown by finding for any algorithm a 'bad' distribution.

Theorem - Lower bound, realizable case. *Let H be a hypothesis set with VC dimension $d > 1$. Then, for any $m \geq 1$ and any learning algorithm $\mathcal{A}$, there exist a distribution $\mathcal{D}$ over X and a target function $f \in H$ such that*

$$\underset{S \sim \mathcal{D}}{P}\left(\mathcal{R}(h_S, f) > \frac{d-1}{32m}\right) \geq 0.01.$$

Hence, for any algorithm $\mathcal{A}$, there exists a 'bad' distribution over X and a target function f for which the error of the hypothesis returned by $\mathcal{A}$ is a multiple of $\frac{d}{m}$ with some constant probability.

The result implies in particular that PAC learning in the realizable case is not possible when the VC dimension is infinite - further demonstrating the key role of VC dimension in learning.

VC dimension

Theorem - Lower bound, non-realizable case. *Let H be a hypothesis set with VC dimension $d > 1$. Then, for any $m \geq 1$ and any learning algorithm $\mathcal{A}$, there exist a distribution $\mathcal{D}$ over $X \times \{0,1\}$ such that*

$$\underset{S \sim \mathcal{D}}{P}\left(\mathcal{R}(h_S) - \inf_{h \in H} \mathcal{R}(h) > \sqrt{\frac{d}{320m}}\right) \geq 0.01.$$

Equivalently, for any learning algorithm, the sample complexity verifies

$$m \geq \frac{d}{320\epsilon^2}$$

Hence, for any algorithm $\mathcal{A}$, there exists a 'bad' distribution over $X \times \{0,1\}$ for which the error of the hypothesis returned by $\mathcal{A}$ is a multiple of $\sqrt{\frac{d}{m}}$ with some constant probability.
In particular, with an infinite VC dimension, agnostic PAC learning is not possible.

VC dimension

We showed that, given any $\delta > 0$, with probability at least $1 - \delta$ and for all $h \in H$, if h is consistent, then

$$\mathcal{R}(h) = O(\frac{1}{m}(\log |H| + \log \frac{1}{\delta})) = O(\frac{1}{m}(\log Gr_H(m) + \log \frac{1}{\delta}))$$

and $m \geq \frac{1}{\epsilon}(\log |H| + \log \frac{1}{\delta})$

Additionally, we observed that if $VC\dim(H) = d$, then $Gr_H(m) = O(m^d)$ and $\log Gr_H(m) = O(d \log m)$. Hence we have:

Proposition

Let $VC\dim(H) = d$. For all consistent $h \in H$, given any $\delta > 0$, with probability at least $1 - \delta$, we have that

$$\mathcal{R}(h) = O(\frac{1}{m}(d \log m + \log \frac{1}{\delta})) = O(\frac{d}{m} \log m - \frac{1}{m} \log \delta)$$

Model Selection

A key problem in the design of learning algorithms:

How should the hypothesis set H be chosen?

This is known as the **model selection problem.**

A rich or complex enough hypothesis set could contain the ideal classifier.

On the other hand, learning with such a complex family becomes a very difficult task.

More generally, the choice of H is subject to a trade-off that can be analyzed in terms of the *estimation and approximation errors*.

Model Selection

In the most general scenario of supervised learning, the training data is a labeled sample S drawn i.i.d. according to a distribution $\mathcal{D}$ defined over $X \times Y$, where X is the input space and Y is the set of labels.

The learning problem is to find a hypothesis $h \in H$ with small generalization error

$$\mathcal{R}(h) = \underset{(x,y)\sim\mathcal{D}}{P}(h(x) \neq y)$$

This general scenario is referred to as the **stochastic scenario.** Within this setting, *the output label is a probabilistic function of the input* and captures many real-world problems where the label of an input point is not unique.

The natural extension of the PAC-learning framework to this setting is known as the **agnostic PAC-learning.**

Model Selection

As above, let n be a number such that the computational cost of representing any element x is at most $O(n)$ and denote by $size(c)$ the maximal cost of the computational representation of $c \in \mathcal{C}$.

Definition (Agnostic PAC-learning)

Let $\mathcal{C}$ be a concept class and H a hypothesis set. $\mathcal{A}$ is an agnostic PAC-learning algorithm if there exists a polynomial function q such that for any $\epsilon > 0$ and $\delta > 0$, for all distributions $\mathcal{D}$ on $X \times Y$, and for any target concept c the following holds for any sample size $m > q(\frac{1}{\epsilon}, \frac{1}{\delta}, n, size(c))$:

$$\underset{S \sim \mathcal{D}^m}{P}(\mathcal{R}(h_S) - \min_{h \in H} \mathcal{R}(h) \leq \epsilon) \geq 1 - \delta$$

If further $\mathcal{A}$ runs in polynomial time $q(\frac{1}{\epsilon}, \frac{1}{\delta})$ then it is said to be an efficient agnostic PAC-learning algorithm.

Model Selection

When the label of a point can be uniquely determined by some measurable function $f : X \to Y$ (with probability one), then the scenario is said to be **deterministic.** In that case, it suffices to consider a distribution $\mathcal{D}$ over the input space.

The training sample is obtained by drawing $(x_1, \ldots, x_m)$ according to $\mathcal{D}$ and the labels are obtained via $f : y_i = f(x_i)$ for all $i \in [m]$. Many learning problems can be formulated within this deterministic scenario.

In the deterministic case, by definition, there exists a target function f with no generalization error: $\mathcal{R}(h) = 0$.

In the stochastic case, there is a minimal non-zero error for any hypothesis.

Model Selection

Definition (Bayes error)

Given a distribution $\mathcal{D}$ over $X \times Y$, the **Bayes error** $\mathcal{R}^*$ is defined as the infimum of the errors achieved by measurable functions $h : X \to Y$:

$$\mathcal{R}^* = \inf_{h \text{ measurable}} \mathcal{R}(h)$$

A hypothesis h with $\mathcal{R}(h) = \mathcal{R}^*$ is called a **Bayes hypothesis** or **Bayes classifier.**

By definition, in the deterministic case, we have $\mathcal{R}^* = 0$, but, in the stochastic case, $\mathcal{R}^* \neq 0$.

Model Selection

The Bayes classifier h_{Bayes} can be defined in terms of the conditional probabilities as

$$h_{Bayes}(x) = \operatorname*{argmax}_{y \in \{0,1\}} P(y|x)$$

The average error made by $h_{Bayes}(x)$ on $x \in X$ is thus $\min\{P(0|x), P(1|x)\}$, and this is the minimum possible error. This quantity is called the **noise** associated with $\mathcal{D}$ and is precisely the Bayes error.

Model Selection

Let H be a family of functions mapping X to $\{-1, +1\}$.
The **excess error** of a hypothesis h chosen from H, that is the difference between its error $\mathcal{R}(h)$ and the Bayes error $\mathcal{R}^*$, can be decomposed as follows

$$\mathcal{R}(h) - \mathcal{R}^* = \left(\mathcal{R}(h) - \inf_{h \in H} \mathcal{R}(h)\right) + \left(\inf_{h \in H} \mathcal{R}(h) - \mathcal{R}^*\right)$$

- The first term is the **estimation error**: It depends on the hypothesis h selected and measures the error of h with respect to the infimum of the errors achieved by hypotheses in H.
- The second term is the **approximation error**: It measures how well the Bayes error can be approximated using H. It is a property of the hypothesis set H - a measure of its richness.

Model Selection

Model selection consists of choosing the hypothesis space H with a favorable trade-off between approximation and estimation errors.

Note: The approximation error is not accessible, since in general the underlying distribution $\mathcal{D}$ needed to determine $\mathcal{R}^*$ is not known.

In contrast, the estimation error of an algorithm $\mathcal{A}$, that is, the estimation error of the hypothesis h_S returned after training on a sample S, can sometimes be bounded using generalization bounds.

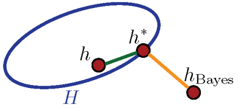


Figure: Estimation error (in green) and approximation error (in orange). Here, we assume that there exists a best-in-class hypothesis $h^* \in H$ such that $\mathcal{R}(h^*) = \inf_{h \in H} \mathcal{R}(h)$

Empirical risk minimization (ERM)

A standard algorithm for which the estimation error can be bounded is **Empirical Risk Minimization (ERM)**.

ERM seeks to minimize the error on the training sample:

$$h_S^{ERM} = \operatorname*{argmin}_{h \in H} \hat{\mathcal{R}}_S(h)$$

Proposition

For any sample S,

$$P\left(\mathcal{R}(h_S^{ERM}) - \inf_{h \in H} \mathcal{R}(h) > \epsilon\right) \leq P\left(\sup_{h \in H} |\mathcal{R}(h) - \hat{\mathcal{R}}_S(h)| > \frac{\epsilon}{2}\right)$$

Empirical risk minimization (ERM)

Proof. By the definition of infimum, for any $\epsilon > 0$ there exists $h_\epsilon \in H$ such that $\mathcal{R}(h_\epsilon) < \inf_{h \in H} \mathcal{R}(h) + \epsilon$, thus $\hat{\mathcal{R}}_S(h_S^{ERM}) \leq \hat{\mathcal{R}}_S(h_\epsilon)$. It follows that

$$\begin{aligned}
\mathcal{R}(h_S^{ERM}) - \inf_{h \in H} \mathcal{R}(h) &= \mathcal{R}(h_S^{ERM}) - \mathcal{R}(h_\epsilon) + \mathcal{R}(h_\epsilon) - \inf_{h \in H} \mathcal{R}(h) \\
&\leq \mathcal{R}(h_S^{ERM}) - \mathcal{R}(h_\epsilon) + \epsilon \\
&= \mathcal{R}(h_S^{ERM}) - \hat{\mathcal{R}}_S(h_S^{ERM}) + \hat{\mathcal{R}}_S(h_S^{ERM}) - \mathcal{R}(h_\epsilon) + \epsilon \\
&\leq \mathcal{R}(h_S^{ERM}) - \hat{\mathcal{R}}_S(h_S^{ERM}) + \hat{\mathcal{R}}_S(h_\epsilon) - \mathcal{R}(h_\epsilon) + \epsilon \\
&\leq 2 \sup_{h \in H} |\mathcal{R}(h) - \hat{\mathcal{R}}_S(h)| + \epsilon
\end{aligned}$$

Since this holds for any $\epsilon > 0$, it follows that

$$\mathcal{R}(h_S^{ERM}) - \inf_{h \in H} \mathcal{R}(h) \leq 2 \sup_{h \in H} |\mathcal{R}(h) - \hat{\mathcal{R}}_S(h)| \qquad \square$$

Empirical risk minimization (ERM)

The right-hand side in the Proposition can be upper-bounded using the generalization bounds in terms of the Rademacher complexity, the growth function, or the VC-dimension.
For instance, we have

$$P\left(\mathcal{R}(h_S^{ERM}) - \inf_{h\in H}\mathcal{R}(h) > \epsilon\right) \leq 2e^{-2m(\epsilon-\mathfrak{R}_m(H))^2}$$

When H admits a favorable Rademacher complexity, e.g., finite VC dimension (in which case $\mathfrak{R}_m(H) = O(\sqrt{\frac{d}{m}})$, for a sufficiently large sample, with high probability, the estimation error is guaranteed to be small.

Nevertheless, the performance of ERM is typically very poor.

1. The ERM approach disregards the complexity of H: either H is not complex enough, in which case the approximation error can be very large, or H is very rich, in which case the bound on the estimation error becomes very loose.
2. In many cases, determining the ERM solution is computationally intractable.

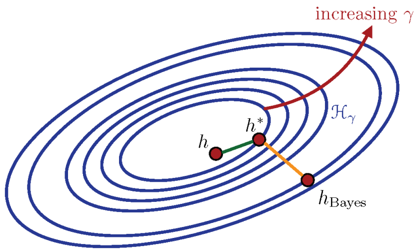
Structural risk minimization (SRM)

The hypothesis set H may be too rich for generalization bounds to hold.

To circumvent this problem, we can decompose H as a union of increasingly complex hypothesis sets:

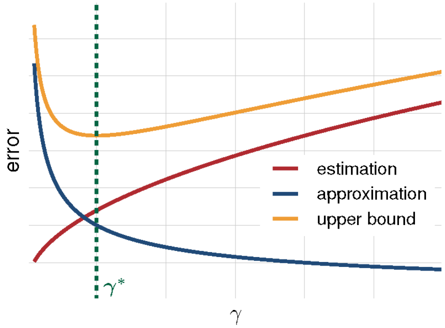
$$H = \bigcup_{\gamma \in \Gamma} H_\gamma$$

with the complexity of H_γ increasing with $\gamma \in \Gamma$.



Structural risk minimization (SRM)

The task is to find γ^* and H_{γ^*} with the most favorable trade-off between estimation and approximation errors.

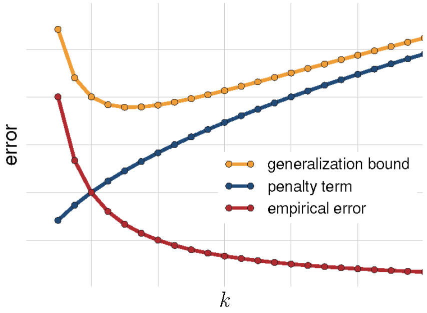


Structural risk minimization (SRM)

Structural Risk Minimization (SRM) method:

1. H is assumed to be decomposable into a countable set $H = \bigcup_{k \in \mathbb{N}} H_k$
2. $H_k \subset H_{k+1}$, $k \in \mathbb{N}$.

SRC seeks to determine the index $k^* \in \mathbb{N}$ and the ERM hypothesis $h \in H_{k^*}$ that minimize an upper bound on the excess error $\mathcal{R}(h) - \mathcal{R}^*$.



Structural risk minimization (SRM)

As we show below, the SRM solution h_S^{SRM} is given by

$$h_S^{SRM} = \underset{h \in H_k, k \in \mathbb{N}}{\operatorname{argmin}} F_k(h) = \underset{h \in H_k, k \in \mathbb{N}}{\operatorname{argmin}} \left(\hat{\mathcal{R}}_S(h) + \mathfrak{R}_m(H_k) + \sqrt{\frac{\log k}{m}} \right)$$

SRM identifies an optimal index k^* and therefore an hypothesis set H_{k^*}, and returns the ERM solution based on that hypothesis set.

For any $h \in H$, we denotes by $H_{k(h)}$ the least complex hypothesis set among the H_k that contain h.

Structural risk minimization (SRM)

Here is the precise statement ensuring

Theorem (SRM Learning guarantee)

For any $\delta > 0$, with probability at least $1 - \delta$ over the draw of an i.i.d. sample S of size m from $\mathcal{D}^m$, the generalization error of the hypothesis h_S^{SRM} returned by the SRM method is bounded by

$$\mathcal{R}(h_S^{SRM}) \leq \inf_{h \in H} \left(\mathcal{R}(h) + 2\mathfrak{R}_m(H_{k(h)}) + \sqrt{\frac{\log k(h)}{m}} \right) + \sqrt{\frac{2 \log \frac{3}{\delta}}{m}}$$

Structural risk minimization (SRM)

Proof.

$$
\begin{aligned}
& P\left(\sup_{h\in H} \mathcal{R}(h) - F_{k(h)}(h)) > \epsilon\right) \\
&= P\left(\sup_{h\in H_k, k\in\mathbb{N}} \mathcal{R}(h) - F_k(h) > \epsilon\right) \\
&\leq \sum_{k=1}^{\infty} P\left(\sup_{h\in H_k} \mathcal{R}(h) - F_k(h) > \epsilon\right) \\
&= \sum_{k=1}^{\infty} P\left(\sup_{h\in H_k} \mathcal{R}(h) - \hat{\mathcal{R}}_S(h) - \mathfrak{R}_m(H_k) > \epsilon + \sqrt{\frac{\log k}{m}}\right) \\
&\leq \sum_{k=1}^{\infty} \exp\left(-2m(\epsilon + \sqrt{\frac{\log k}{m}})^2\right) \\
&\leq \sum_{k=1}^{\infty} e^{-2m\epsilon^2} e^{-2\log k} \\
&\leq e^{-2m\epsilon^2} \sum_{k=1}^{\infty} \frac{1}{k^2} \leq 2e^{-2m\epsilon^2}
\end{aligned}
$$

Structural risk minimization (SRM)

We observe that, for any two random variables X_1, X_2 with $X_1 + X_2 > \epsilon$, then either $X_1 > \frac{\epsilon}{2}$ or $X_2 > \frac{\epsilon}{2}$. Thus $P(X_1 + X_2 > \epsilon) \leq P(X_1 > \frac{\epsilon}{2}) + P(X_2 > \frac{\epsilon}{2})$. Using this observation, the inequality derived above and the inequality $F_{k(h_S^{SRM})}(h_S^{SRM}) \leq F_{k(h)}(h)$, we have

$$\begin{aligned}
&P\left(\mathcal{R}(h_S^{SRM}) - \mathcal{R}(h) - 2\mathfrak{R}_m(H_{k(h)}) - \sqrt{\frac{\log k(h)}{m}} > \epsilon\right) \\
&\leq P\left(\mathcal{R}(h_S^{SRM}) - F_k(k(h) > \frac{\epsilon}{2}\right) \\
&+ P\left(F_{k(h)}(h) - \mathcal{R}(h) - 2\mathfrak{R}_m(H_{k(h)}) - \sqrt{\frac{\log k(h)}{m}} > \tfrac{\epsilon}{2}\right) \\
&\leq 2e^{-\frac{m\epsilon^2}{2}} + P\left(F_{k(h)}(h) - \mathcal{R}(h) - 2\mathfrak{R}_m(H_{k(h)}) - \sqrt{\frac{\log k(h)}{m}} > \tfrac{\epsilon}{2}\right) \\
&= 2e^{-\frac{m\epsilon^2}{2}} + P\left(\hat{\mathcal{R}}_S(h) - \mathcal{R}(h) - \mathfrak{R}_m(H_{k(h)}) > \tfrac{\epsilon}{2}\right) \\
&= 2e^{-\frac{m\epsilon^2}{2}} + e^{-\frac{m\epsilon^2}{2}} = 3e^{-\frac{m\epsilon^2}{2}}.
\end{aligned}$$

The proof is completed by setting the RHS to be equal to δ. $\square$

Structural risk minimization (SRM)

Implications of the SRM Learning guarantee Theorem.

For simplicity, let us assume that there exists h^* such that $\mathcal{R}(h^*) = \inf_{h \in H} \mathcal{R}(h)$.
Then the theorem implies that, with probability at least $1 - \delta$, for all $h \in H$

$$\mathcal{R}(h_S^{SRM}) \leq \mathcal{R}(h^*) + 2\mathfrak{R}_m(H_{k(h)}) + \sqrt{\frac{\log k(h^*)}{m}} + \sqrt{\frac{2 \log \frac{3}{\delta}}{m}}$$

The guarantee for SRM is as favorable as the one we would have obtained, had an oracle informed us of the index $k(h^*)$ of the best-in-class classifier's hypothesis set.

When H is rich enough that $\mathcal{R}(h^*)$ is close to the Bayes error, the learning bound of the theorem is approximately a bound on the excess error of the SRM solution.

Structural risk minimization (SRM)

SRM has drawbacks.

The decomposability of H into countably many hypothesis sets, each with a converging Rademacher complexity, is a strong assumption.
As an example, the family of all measurable functions cannot be written as a union of countably many hypothesis sets with finite VC-dimension.
Thus, the choice of H or that of the hypothesis sets H_k is a key component of SRM.

SRM is typically computationally intractable: for most hypothesis sets, finding the solution of ERM is NP-hard and in general SRM requires determining that solution for a large number of indices k.

Cross-validation

An alternative method for model selection is cross-validation.

Cross-validation (CV) consists of using some fraction of the training sample as a validation set to select a hypothesis set H_k.

This is in contrast with SRM which relies on a theoretical learning bound assigning a penalty to each hypothesis set.

The CV algorithm is organized as follows.

• Let $S \in X \times Y$ be an i.i.d. labeled sample of size m.
We divide

$$S = S_1 \cup S_2$$

where

S_1 of size $(1 - \alpha)m$ is reserved for **training**;

S_2 of size αm is reserved for **validation**.

Here $\alpha \in (0, 1)$ typically chosen to be relatively small.

Cross-validation

- Let (H_k) be a countable sequence of hypothesis sets with increasing complexities.
- For any $k \in \mathbb{N}$, let $h^{ERM}_{S_1,k}$ denote the solution of ERM run on S_1 using the hypothesis set H_k.
- The hypothesis h^{CV}_S returned by cross-validation is the ERM solution $h^{ERM}_{S_1,k}$ with the best performance on S_2

$$h^{CV}_S = \operatorname*{argmin}_{h \in \{h^{ERM}_{S_1,k}, k \in \mathbb{N}\}} \hat{\mathcal{R}}_{S_2}(h)$$

Proposition

For any $\alpha > 0$ and any sample size $m \geq 1$,

$$P\left(\sup_{k \in \mathbb{N}} |\mathcal{R}(h^{ERM}_{S_1,k}) - \hat{\mathcal{R}}_{S_2}(h^{ERM}_{S_1,k})| > \epsilon + \sqrt{\frac{\log k}{\alpha m}}\right) \leq 4\, e^{-2\alpha m \epsilon^2}$$

Cross-validation

Proof.

$$P\left(\sup_{k\in\mathbb{N}} |\mathcal{R}(h^{ERM}_{S_1,k}) - \hat{\mathcal{R}}_{S_2}(h^{ERM}_{S_1,k})| > \epsilon + \sqrt{\frac{\log k}{\alpha m}}\right)$$
$$\leq \sum_{k=1}^{\infty} P\left(|\mathcal{R}(h^{ERM}_{S_1,k}) - \hat{\mathcal{R}}_{S_2}(h^{ERM}_{S_1,k})| > \epsilon + \sqrt{\frac{\log k}{\alpha m}}\right)$$
$$= \sum_{k=1}^{\infty} E\left(P\left(|\mathcal{R}(h^{ERM}_{S_1,k}) - \hat{\mathcal{R}}_{S_2}(h^{ERM}_{S_1,k})| > \epsilon + \sqrt{\frac{\log k}{\alpha m}}\,\middle|\, S_1\right)\right)$$

Since he hypothesis $h^{ERM}_{S_1,k}$ is fixed conditioned on S_1 and the sample S_2 is independent from S_1, by Hoeffding's inequality,

$$P\left(|\mathcal{R}(h^{ERM}_{S_1,k}) - \hat{\mathcal{R}}_{S_2}(h^{ERM}_{S_1,k})| > \epsilon + \sqrt{\frac{\log k}{\alpha m}}\,\middle|\, S_1\right) \leq 2\, e^{-2\alpha m(\epsilon + \sqrt{\frac{\log k}{\alpha m}})^2}$$
$$\leq 2\, e^{-2\alpha m \epsilon^2 - 2\log k}$$
$$\leq \frac{2}{k^2}\, e^{-2\alpha m \epsilon^2}$$

Cross-validation

From above:

$$P\left(|\mathcal{R}(h^{ERM}_{S_1,k}) - \hat{\mathcal{R}}_{S_2}(h^{ERM}_{S_1,k})| > \epsilon + \sqrt{\frac{\log k}{\alpha m}} \,\middle|\, S_1\right) \leq \tfrac{2}{k^2}\, e^{-2\alpha m \epsilon^2}$$

Using this inequality in the above bound and summing over $k \in \mathbb{N}$:

$$\begin{aligned} P\left(\sup_{k\in\mathbb{N}} |\mathcal{R}(h^{ERM}_{S_1,k}) - \hat{\mathcal{R}}_{S_2}(h^{ERM}_{S_1,k})| > \epsilon + \sqrt{\frac{\log k}{\alpha m}}\right) &\leq \sum_{k=1}^{\infty} \tfrac{2}{k^2}\, e^{-2\alpha m \epsilon^2} \\ &\leq 4\, e^{-2\alpha m \epsilon^2}. \qquad \square \end{aligned}$$

Cross-validation

Let $\mathcal{R}(h^{SRM}_{S_1,k})$ be the generalization error of the SRM solution using a sample S_1 of size $(1-\alpha)m$ and $\mathcal{R}(h^{CV}_S)$ the generalization error of the cross-validation solution using a sample S of size m.

Then, using the above proposition, we derive the following learning guarantee which compares the error of CV vs SRM methods.

Theorem - CV versus SRM

For any $\delta > 0$, with probability at least $1-\delta$ over the draw of an i.i.d. sample S of size m from $\mathcal{D}^m$,

$$\mathcal{R}(h^{CV}_S) - \mathcal{R}(h^{SRM}_{S_1}) \leq 2\sqrt{\frac{\log \max(k(h^{CV}_S), k(h^{SRM}_{S_1}))}{\alpha m}} + 2\sqrt{\frac{\log \frac{4}{\delta}}{2\alpha m}}$$

where, for any h, $k(h)$ denotes the smallest index of a hypothesis set containing h.

Cross-validation

Proof. Using the above proposition, the Theorem for SRM Learning guarantee and the property that h_S^{CV} is a minimizer, $\delta > 0$, with probability at least $1-\delta$, we have

$$\begin{aligned}
\mathcal{R}(h_S^{CV}) &\le \hat{\mathcal{R}}_{S_2}(h_S^{CV}) + \sqrt{\frac{\log k(h_S^{CV})}{\alpha m}} + \sqrt{\frac{\log \frac{4}{\delta}}{2\alpha m}} \\
&\le \hat{\mathcal{R}}_{S_2}(h_{S_1}^{SRM}) + \sqrt{\frac{\log k(h_S^{CV})}{\alpha m}} + \sqrt{\frac{\log \frac{4}{\delta}}{2\alpha m}} \\
&\le \mathcal{R}(h_{S_1}^{SRM}) + \sqrt{\frac{\log k(h_S^{CV})}{\alpha m}} + \sqrt{\frac{\log k(h_{S_1}^{SRM})}{\alpha m}} + 2\sqrt{\frac{\log \frac{4}{\delta}}{2\alpha m}} \\
&\le \mathcal{R}(h_{S_1}^{SRM}) + 2\sqrt{\frac{\log \max(k(h_S^{CV}), k(h_{S_1}^{SRM}))}{\alpha m}} + 2\sqrt{\frac{\log \frac{4}{\delta}}{2\alpha m}} \qquad \square
\end{aligned}$$

Cross-validation

Interpretation:

The learning guarantee shows that, with high probability, *the generalization error of the CV solution for a sample of size m is close to that of the SRM solution for a sample of size $(1 - \alpha)m$.*

For α relatively small, this suggests a guarantee similar to that of SRM, which, as previously discussed, is very favorable.

However, in some unfavorable regimes, an SRM algorithm trained on $(1 - \alpha)m$ points may have a significantly worse performance than when trained on m points

Thus, the bound suggests in fact a trade-off: α should be chosen sufficiently small to avoid the unfavorable regimes just mentioned and yet sufficiently large for the right-hand side of the bound to be small and thus informative.

Cross-validation

n-**fold cross-validation** is a method used in practice *to correct issues arising when m is small.*

The method consists of first randomly partitioning a given sample S of m labeled examples into n subsamples, or folds.
Then, for any $i \in [n]$, the learning algorithm is trained on all but the i-th fold to generate a hypothesis h_i, and the performance of h_i is tested on the i-th fold.

The parameter value of the algorithm is evaluated based on the average error of the hypotheses h_i, which is called the **cross-validation error**.

The special case of n-fold cross-validation where $n = m$ is called **leave-one-out cross-validation**, since at each iteration exactly one instance is left out of the training sample.

Regularization

Another broad family of algorithms used to control the generalization error is that of **regularization**-based algorithms.

This approach is inspired by the SRM method where the hypothesis space H is decomposed as a union of increasingly complex hypothesis sets:

$$H = \bigcup_{\gamma \in \Gamma} H_\gamma$$

with the complexity of H_γ increasing with $\gamma \in \Gamma$.

Regularization

We consider a very complex space of hypotheses H that is represented as an uncountable union of nested hypothesis sets

$$H = \bigcup_{\gamma} H_{\gamma}$$

H is often chosen to be dense in the space of continuous functions over X.
For instance, H may be chosen to be the set of all linear functions in some high-dimensional space and

$$H_{\gamma} = \{x \mapsto w \cdot \Phi(x) : \|w\| \leq \gamma\}$$

Given a labeled sample $S \subset X \times Y$, *the extension of the SRM method to an uncountable union leads to a* **regularized solution**.

Regularization

Recall that the SRM solution h_S^{SRM} is associated with the optimization problem

$$h_S^{SRM} = \operatorname*{argmin}_{h \in H_k, k \in \mathbb{N}} \left(\hat{\mathcal{R}}_S(h) + \mathfrak{R}_m(H_k) + \sqrt{\frac{\log k}{m}} \right)$$

To deal with an uncountable union, we are led to consider

$$\operatorname*{argmin}_{h \in H_\gamma, \gamma > 0} \left(\hat{\mathcal{R}}_S(h) + \mathfrak{R}_m(H_\gamma) + \sqrt{\frac{\log \gamma}{m}} \right)$$

where another penalty terms $p(\gamma, m)$ can be chosen in place of $p(\gamma, m) = \mathfrak{R}_m(H_\gamma) + \sqrt{\frac{\log \gamma}{m}}$ leading to

$$\operatorname*{argmin}_{h \in H_\gamma, \gamma > 0} \left(\hat{\mathcal{R}}_S(h) + p(\gamma, m) \right)$$

Regularization

Under some conditions, there exists a function $r : H \to \mathbb{R}$ such that, for any $\gamma > 0$, the constrained optimization problem

$$\underset{h \in H_\gamma, \gamma > 0}{\operatorname{argmin}} \ \hat{\mathcal{R}}_S(h) + p(\gamma, m)$$

can be written as the unconstrained optimization problem

$$\underset{h \in H}{\operatorname{argmin}} \hat{\mathcal{R}}_S(h) + \lambda r(h),$$

for some $\lambda > 0$.

$r(h)$ is called a **regularization term** and the **regularization parameter** λ is treated as a hyperparameter since its optimal value is typically not known.

Larger values of λ further penalize more complex hypotheses, while, for λ close or equal to zero, the regularization term has no effect and the algorithm coincides with ERM.

Regularization

For most algorithms, the regularization term $r(h)$ is chosen to be an increasing function of $\|h\|$ for some choice of the norm $\|\|$, when H is the subset of a Hilbert space.

When the regularization term is chosen to be $\|h\|_p$ for some choice of $p \geq 1$, then it is a convex function of h, since any norm is convex.

However, for the zero-one loss, the first term of the objective function is non-convex, thereby making the optimization problem computationally hard.

In practice, most regularization-based algorithms instead *use a convex upper bound* on the zero-one loss and replace the empirical zero-one term with the empirical value of that convex surrogate. The resulting optimization problem is then convex and therefore admits more efficient solutions than SRM.

Regularization

• Regularization provides a unified view on many machine learning methods.

By plugging in different regularizers and (surrogate) loss functions, we obtain different machine learning models.

For instance, in the standard SVM formulation, training means to learn the model parameter vector w by solving the optimization problem

$$\min_{w} \mathcal{L}(w) = \min_{w} \frac{1}{m} \sum_{i=1}^{m} L(y_i, h_w(x_i)) + \lambda \|w\|_2^2$$

where $L(y_i, h_w(x_i)) = \max[1 - y_i w^t(x_i + b), 0]$ is the **hinge loss**.

SVM uses the hinge loss as error measure and the ℓ^2-regularizer to penalize complex solutions.

Decision Theoretic Generalizations of the PAC Model

Limitations of the classical PAC learning model

- The model is usually defined for binary functions. It is useful in practice to consider more general functions.
- The assumption that the examples are generated from an underlying target function is too restrictive in some applications where one would like a more general regression model in which the y component in a training example $(x, y) \in X \times Y$ follows a conditional distribution on Y given x Here the goal is to approximate this conditional distribution.
- Many learning problems are unsupervised and the learner has access only to randomly drawn unlabeled examples from an instance space X. Here learning can be viewed as some form of approximation of the distribution that is generating these examples. This is called **density estimation** when the instance space X is continuous and no specific parametric form for the underlying distribution on X is assumed; it is called **parameter estimation** when specific parametric probability models are used.

Decision Theoretic Generalizations of the PAC Model

To extend the PAC model, one can apply a more general framework based on statistical decision theory.

In this general framework we assume the learner receives randomly drawn training examples each example consisting of an instance $x \in X$ and and outcome $y \in Y$ where X and Y are arbitrary sets called instance and outcome spaces respectively. These examples are generated according to an unknown joint distribution on $X \times Y$.

After training, the learner will receive further random examples drawn from this same joint distribution. For each example (x, y) the learner will be shown only the instance x. Then he will be asked to choose an action a from a set of possible actions A called the **decision space**.

Decision Theoretic Generalizations of the PAC Model

Following the action by the learner, the outcome y will be revealed to the learner.

In the case that we examine here the outcome y depends only on the instance x and not on the action a chosen by the learner. For each action a and outcome y, the learner will suffer a **loss** which is measured by a fixed real-valued loss function L on $Y \times A$. We assume that the loss function is known to the learner. The learner tries to choose his actions so as to minimize his loss.

Based on the training examples, the learner develops a deterministic strategy that specifies what he believes is the appropriate action a for each instance $x \in X$. The learner then uses this strategy on all future examples. Thus we look at **batch learning** rather than **incremental** or **on-line** learning.

Decision Theoretic Generalizations of the PAC Model

The learner's strategy, which is a function from the instance space X into the decision space A is called a **decision rule**.

We assume that the decision rule is chosen from a fixed decision rule space H of functions from X into A.
For example, instances in X may be encoded as inputs to a neural network and outputs of the network may be interpreted as actions in A.
In this case, the network represents a decision rule and the decision rule space H may be all functions represented by networks obtained by varying the parameters of a fixed underlying network.

The goal of learning is to find a decision rule in H that minimizes the expected loss when examples are drawn at random from the unknown joint distribution on $X \times Y$

Decision Theoretic Generalizations of the PAC Model

Example

We consider the problem of learning to maximize profit (or to minimize loss) at the horse races.

Here an instance $x \in X$ is a race, an action $a \in A$ consists of placing or not placing a certain bet and an outcome $y \in Y$ is determined by the winner and the second and third place finishers.

The loss $L(y, a)$ is the amount of money lost when bet a is placed and the outcome of the race is y. A negative loss is interpreted as gain

The joint distribution on $X \times Y$ represents the probability of various races and outcomes and is unknown to the learner.
We only have random example $(x_1, y_1), \ldots, (x_n, y_n)$ each consisting of a race outcome pair generated from this distribution.

Decision Theoretic Generalizations of the PAC Model

From these examples, the learner develops a deterministic betting strategy **decision rule**.

The best decision rule h is one that specifies a bet a for each race x that minimizes the expectation of the loss $L(x, a)$ when y is chosen randomly from the unknown conditional distribution on Y given x which is determined by the underlying joint distribution on $X \times Y$.

This is (not necessarily unique) best decision rule minimizes the expected loss on a random example (x, y). This is known as the **Bayes optimal decision rule**.

The learner tries to approximate Bayes optimal decision rule as best he can using decision rules from a given decision rule space H Decision rules that can be represented by a particular kind of neural network.

Decision Theoretic Generalizations of the PAC Model

Other examples include: classification, regression, density and parameter estimations, which are associated with different types of loss functions.

There are three major practical issues in this decision theoretic view of learning.

- The first is the number of random examples needed in order to be able to produce a good decision rule in the decisiojn rule space H.
- The second is the adequacy of the decision rule space H
- The third practical problem is the computational complexity of the method we use to produce our decision rule from the training examples.

Cf. *Overview of the Probably Approximately Correct PAC Learning Framework*, by David Haussler