MATH 6397 - Mathematics of Data Science From signal processing to Convolutional Neural Networks

Instructor: Demetrio Labate

March 11, 2021

Course Outline

1. Mathematics of signal processing
   1.1 Fourier series
   1.2 Wavelets
   1.3 Shearlets
   1.4 Wavelet Scattering Transform
2. Mathematics of machine learning
   2.1 Geometry of high dimensional data
   2.2 Statistical learning theory
   2.3 Support Vector Machines
   2.4 Convolutional Neural Networks

References:

- ☐ *The Mathematics of Signal Processing*, by Damelin and Miller
- ☐ *Foundations of Data Science*, by Blum, Hopcroft and Kannan
- ☐ *Foundations of Machine Learning*, by Mohri, Rostamizadeh and Talwalkar
- ☐ *Deep Learning with PyTorch*, by Stevens, Antiga and Viehmann

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.

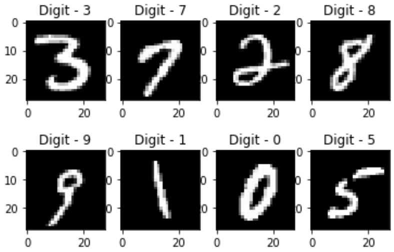


Figure: Handwritten digits and corresponding labels, taken from the MNIST database.

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

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

  where $\theta \in \mathbb{R}^D$ and $\theta_0 \in \mathbb{R}$ 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

More formally, 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 an affine function.
2. **Training or 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.

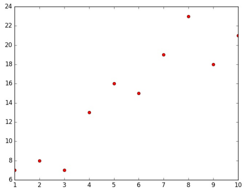
The ability of the trained model to categorize correctly new examples (not part of the training set) is called **generalization.**

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**

$$L(y_n, f(x_n, \theta))$$

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 $S_N = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ 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 n large) the empirical mean is a good estimate of the population mean.

Hence we define the **empirical risk** as 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 $\tilde{x} = (1, x_1, \dots, x_d)^t \in \mathbb{R}^{d+1}$ and similarly any $\theta = (\theta_1, \dots, \theta_d)^t \in \mathbb{R}^d$ to $\tilde{\theta} = (\theta_0, \theta_1, \dots, \theta_d)^t \in \mathbb{R}^{d+1}$.

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

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

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:

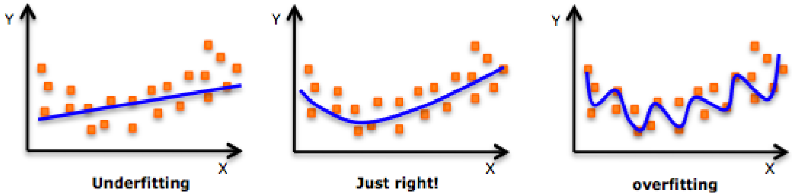
1. How to estimate the expected risk?
2. How to control the expected risk from the empirical risk?

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} underestimates R_{true}.



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 or to reflect other 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)

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

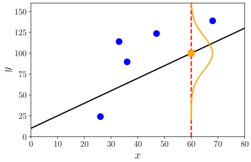


Figure: The uncertainty of the data is described using a Gaussian model.

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:

1. What can be learned efficiently?
2. How many examples are needed to learn successfully?

The **Probably Approximately Correct (PAC)** learning framework defines the class of learnable **concepts** in terms of

1. number of sample points needed to achieve an approximate solution;
2. sample complexity;
3. time and space complexity of the learning algorithm.

The PAC Learning Framework

Notation:

- The **input space** X containing all possible examples or instances.
- 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** is a set of concepts we may wish to learn and is denoted by $\mathcal{C}$.
- 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 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.

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

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 $S = (x_1, \ldots, x_m)$ 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 is equal to the generalization error:

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

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

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

which holds for any x in S. Thus

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

Note that this 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 $O(n)$. In addition, let h_S denote the hypothesis returned by algorithm $\mathcal{A}$ after receiving a labeled sample S.

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 $m > q(\frac{1}{\epsilon}, \frac{1}{\delta}, n, size(c))$:

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

The PAC Learning Framework

A concept class $\mathcal{C}$ is thus PAC-learnable if the hypothesis returned by the algorithm after observing a number of points polynomial in $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$ is approximately correct (error at most ϵ) with high probability (at least $1-\delta$), which justifies the PAC terminology.

$1-\frac{1}{\delta}$ is the **confidence** of the estimate
$1-\epsilon$ is the **accuracy**

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 $\mathcal{D}$ 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

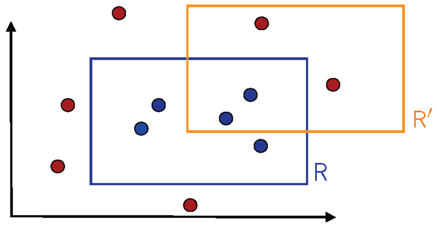
Example: Learning axis-aligned rectangles.
Consider the case where the set of instances are points in the plane $X = \mathbb{R}^2$ and the concept class $\mathcal{C}$ is the set of all axis-aligned rectangles lying in $\mathbb{R}^2$.

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.

We will show that the concept class of axis-aligned rectangles is PAC-learnable

The PAC Learning Framework



$R \in \mathcal{C}$ represents a target axis-aligned rectangle and R' a hypothesis.

The error regions of R' are formed by the area within the rectangle R but outside the rectangle R' (false negatives) and the area within R' but outside the rectangle R (false positives).

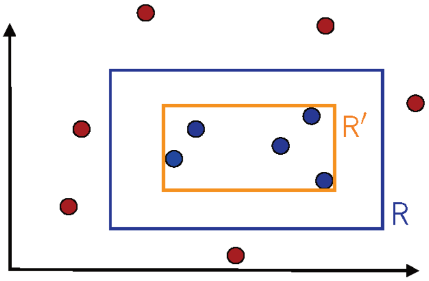
False negatives: points that are labeled as 0 or negatively by R', which are in fact positive or labeled with 1.
False positives: that is, points labeled positively by R' which are in fact negatively labeled.

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.

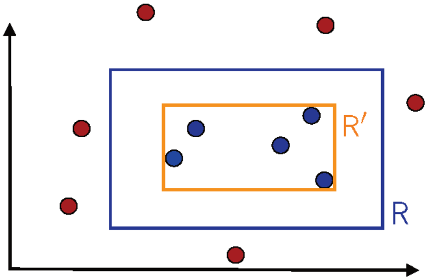
By construction, R_S does not produce any false positives since its points must be included in the target concept R.
Thus, the error region of R_S is included in R.



The PAC Learning Framework

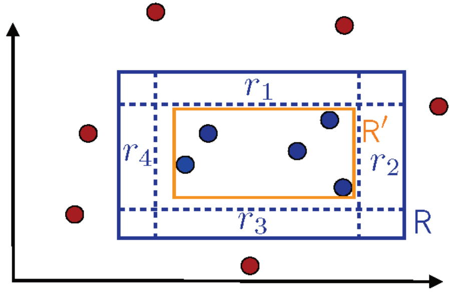
Let $P(R)$ denote the probability mass of the region defined by R, that is the probability that a point randomly drawn according to $\mathcal{D}$ falls within R.

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

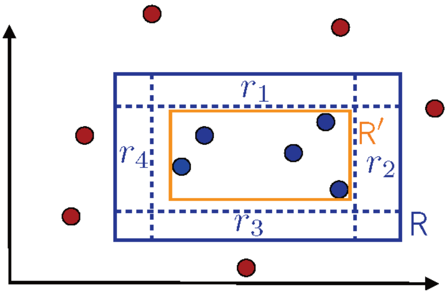
We can define four rectangular regions r_1, r_2, r_3, and r_4 along the sides of R, each with probability at least $\epsilon/4$.
These regions can be constructed by starting with the full rectangle R and then decreasing the size by moving one side as much as possible while keeping a distribution mass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

$$4\,e^{-m\frac{\epsilon}{4}} \leq \delta \iff m \geq \tfrac{4}{\epsilon}\log\tfrac{4}{\delta}$$

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, 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

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 this case, this is obtained by setting δ equal to the bound found above, that is,

$$\delta = 4\, e^{-m\frac{\epsilon}{4}}$$

and solving for ϵ.

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

$$\mathcal{R}(R_S) \leq \tfrac{4}{m} \log \tfrac{4}{\delta}$$

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 is key and 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 i.i.d. sample S returns a consistent hypothesis $h_S : \hat{\mathcal{R}}(h_S) = 0$. 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 $\epsilon, \delta > 0$, with probability at least $1 - \delta$,

$$\mathcal{R}(h_S) \leq \frac{1}{m}(\log |H| + \log \frac{1}{\delta})$$

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.

The PAC Learning Framework

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

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

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 follows 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

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 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 $\delta > 0$, with probability at least $1 - \delta$,

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

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

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

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

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 of the previous chapter are uninformative when dealing with infinite hypothesis sets - which is the typical situation found in machine learning applications.

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

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. 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 independent uniform random variables taking values in ± 1 (= Rademacher r.v.).

Rademacher Complexity

Interpretation.
Let $g_S = (g(x_1, y_1), \ldots, g(x_m, y_m))^t$
Then 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.

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. 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 $\mathcal{D}$:

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

We have our first 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

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

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

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 $\delta > 0$, with probability at least $1 - \delta$,

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

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

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

Let S and S' be two samples differing by exactly one point, say (x_i, y_i). Then, 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

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

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 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

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

Rademacher Complexity

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

Lemma. 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 Lemma.
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. □

Rademacher Complexity

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

Theorem - Rademacher complexity bounds (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

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

and

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

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

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

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

Each one of these distinct classifications is called a dichotomy. Thus, the growth function counts the number of dichotomies that are realized by the hypothesis and this provides a measure of the richness of the hypothesis set H.

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

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

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

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

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 $[-r_i, +r_i]$ 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

We can now bound the Rademacher complexity in terms of the growth function.

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

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

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

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

VC dimension

The computation of the growth function is not always convenient 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 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

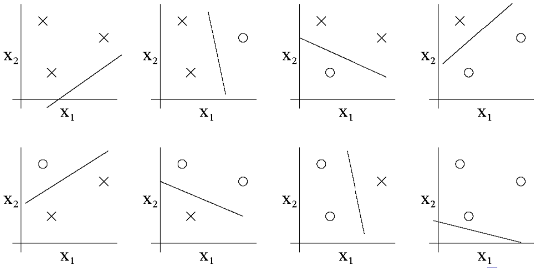
Definition. *The* **VC dimension** *of a hypothesis set H is the size of the largest set that can be shattered by H, that is*

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

Example: Lines in $\mathbb{R}^2$. Let $X = \mathbb{R}^2$. We want to learn $c : X \mapsto \{-1, +1\}$ using the the hypothesis set H of lines in $\mathbb{R}^2$.

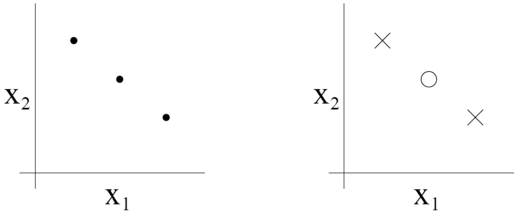
What is the VC dimension of H?

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 $VC\text{dim}(H) = 3$.

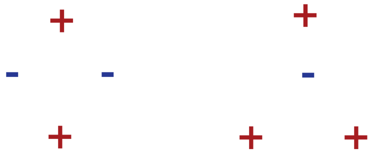
Note. The condition $VC\text{dim}(H) = d$ does not imply that all sets of size d or less are shattered and, in fact, this is typically not the case. It is sufficient that a set of size d is shattered by H.

VC dimension

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

(i) The four points lie on the convex hull defined by the fourpoints, 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 $VC\text{dim}(H) = d + 1$.

Example: Hyperplanes in $\mathbb{R}^d$. Let $X = \mathbb{R}^d$. We want to learn $c : X \mapsto \{-1, +1\}$ using the hypothesis set H consisting of hyperplanes in $\mathbb{R}^d$.

We derive a lower bound of $VC\text{dim}(H)$ by starting with a set of $d + 1$ points in 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 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 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 $i \in J_2$.

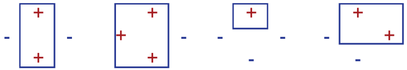
By the definition of convex hull, this implies that $\sum_{i \in J_1} \frac{\beta_i}{\beta} x_i$ belongs to both the convex hull of X_1 and X_2. □

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

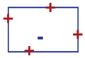
$$conv(X) = \left\{ \sum_{i=1}^{m} \alpha_i x_i : m \geq 1, x_i \in X, \alpha_i \geq 0, \sum_{1}^{m} \alpha_i = 1 \right\}$$

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.

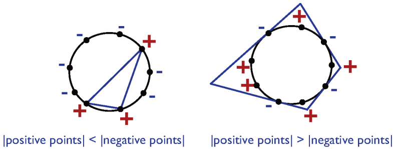


If we assign a negative label to this interior point and a positive label to each of the remaining four points, there is no axis-aligned rectangle that can realize this labeling. Hence, no set of five distinct points can be shattered. Thus $VC\text{dim}(H) = 4$.

VC dimension

Example: Convex d-gons in $\mathbb{R}^2$. We select our hypothesis class to consist of convex polygons with d sides. We will show that it shatters $2d + 1$ points. [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.

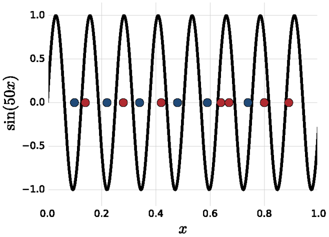

|positive points| < |negative points|
|positive points| > |negative points|

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

VC dimension

The examples presented above could 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

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:*

$$Gr_H(m) \leq \sum_{i=1}^{d} \binom{m}{i}$$

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

Corollary. *Let H be a hypothesis set with $VC\dim(H) = d$. 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 $VC\dim(H) = d < \infty$, in which case $Gr_H(m) = O(m^d)$, or $VC\dim(H) = \infty$, in which case $Gr_H(m) = 2^m$.

VC dimension

Using the relationship between VC dimension and the growth function, we can derive 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 $h \in H$*

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

This shows that the generalization bound is of the form

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

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

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*

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

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*

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

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.

Model Selection

A key problem in the design of learning algorithms is the choice of the hypothesis set H.
This is known as the **model selection problem.**

How should the hypothesis set H be chosen?

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