Foundations

Course of Machine Learning
Master Degree in Computer Science
University of Rome "Tor Vergata"
a.a. 2024-2025

Giorgio Gambosi

The cornerstone of machine learning lies in our access to a collection of data points, each representing the observation of the values of a set of predefined variables, as generated by an unknown process. The crux of this framework is the premise that these data, far from being random, possess an intrinsic structure—albeit one that often proves elusive to identify.

The essence of machine learning lies in two primary objectives, each addressing a different facet of data analysis:

Unsupervised Learning Here, our goal is to extract profound insights into the underlying structure of the data, thereby deepening our understanding of its inherent nature. This approach often involves techniques such as clustering, dimensionality reduction, or generative modeling. A particularly compelling application of unsupervised learning is the ability to algorithmically generate new data points that are, to a high degree, indistinguishable from the original dataset. These synthetic data points appear to be produced by the same unknown process, effectively mimicking the data's intrinsic characteristics and distributions.

Supervised Learning In this paradigm, we aim to predict additional information for each data item based on existing knowledge. This typically involves training models on labeled data to make predictions on unseen instances. Supervised learning encompasses a wide array of tasks, including classification, regression, and sequence prediction.

In both these scenarios, we start with a set of observations already produced by the unknown process. This initial dataset serves as the foundation upon which we build our models and derive our insights.

An alternative framework, distinct from the supervised and unsupervised paradigms, is **reinforcement learning**. In this approach, we posit an interactive relationship with the underlying process. Rather than working with a static dataset, we engage in a dynamic, iterative procedure:

1. At each step, we interact with the process by choosing and performing an action from a set of options
2. as a consequence, we obtain a response under the form of a reward
3. the overarching aim is to identify an optimal strategy for interacting with the unknown process, one that maximizes the cumulative reward.

This framework finds applications in areas such as game playing, robotics, and autonomous systems, where the ability to learn and adapt through trial and error is paramount.

Setting aside the reinforcement learning paradigm, let us delve deeper into the common scenario encountered in supervised and unsupervised learning:

- A training set of n items is represented as a set of input vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$. These vectors encapsulate the features or attributes of each data point and serve as the raw material from which we derive our **model**. The

dimensionality and nature of these vectors can vary widely depending on the specific problem domain, ranging from low-dimensional numerical data to high-dimensional representations of images, text, or complex structures.

- In the case of supervised learning, the training set is augmented with additional information. Specifically, it includes a target vector $\mathbf{t} = \{t_1, \ldots, t_n\}$, where each t_i specifies the value to be predicted on the basis of the corresponding input vector $\mathbf{x}_i$. This pairing of inputs and targets forms the basis for training predictive models that can generalize to unseen data.

Observe that vectors in the training set are in general a specific representation of items (which are real-world entities), just like tuples in a database are just models (according to a predefined data model) of entities. The set of features used in the learning process is a fundamental component in such a process and may have an important effect on the quality and efficiency of the predictions. Adjusting the set of features (by both defining new features from suitable functional compositions of given ones and identifying features which are significant for the task to be performed) is a fundamental issue in Machine learning, which may involve both domain expertise and sophisticated algorithmic approaches, applied in the frameworks of unsupervised learning.

Supervised learning

Our objective is to predict the unknown value of an additional feature, termed the *target*, for a given item $\mathbf{x}$, based on the values of a set of features. This prediction task takes two primary forms:

Regression When the target is a real $t \in \mathbb{R}$

Classification When the target is a discrete value, from a predefined set $t \in \{1, \ldots, K\}$

To achieve this, we employ a general approach that involves defining a (functional or probabilistic) model of the relationship between feature and target values. This model is derived through a learning process from a set of examples, illustrating the relationship between the set of features and the target. The examples are collected in a training set $\mathcal{T} = (\mathbf{X}, \mathbf{t})$, and each example comprises:

- A feature vector $\mathbf{x}_i = \{x_{i1}, \ldots, x_{im}\}$
- The corresponding target value t_i

The model we construct can take one of two forms:

1. A function $y(\cdot)$ which, for any item $\mathbf{x}$, returns a value $y(\mathbf{x})$ as an estimate of t. This function acts as a direct predictor, mapping the input features to the target space.
2. A probability distribution that associates each possible value $\overline{y}$ in the target domain with its corresponding probability $p(y = \overline{y}|\mathbf{x})$. This probabilistic approach provides a more nuanced view, capturing the uncertainty inherent in the prediction task.

The choice between these model types often depends on the specific requirements of the problem at hand, the nature of the data, and the desired interpretability of the results. The function-based approach offers straightforward predictions, while the probabilistic model provides a richer representation of the underlying uncertainties and potential outcomes.

In both cases, the model serves as a bridge between the observed features and the target we aim to predict, leveraging the patterns and relationships learned from the training data to make informed predictions on new, unseen instances.

Unsupervised learning

The objective here is to detect inherent patterns and structures within a given collection of items, known as the dataset $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, where no target values are associated with the items, in order to extract synthetic information such data. The synthetic information we seek to extract can take several forms:

Clustering Identifying subsets of similar items within the dataset. This process involves grouping data points based on their intrinsic similarities, revealing natural structures or segments in the data.

Density Estimation Determining the distribution of items in their domain. This approach aims to model the underlying probability density function that generated the observed data, providing insights into the data's statistical properties and potential generative processes.

Dimensionality Reduction Projecting items onto lower-dimensional subspaces while preserving as much information as possible. This can be achieved through two main approaches:

- Feature Selection: Identifying and retaining the most informative subset of original features.
- Feature Extraction: Creating new, lower-dimensional representations of the data that capture its essential characteristics.

These dimensionality reduction techniques aim to characterize the items using a smaller set of features, potentially revealing latent structures and reducing computational complexity for subsequent analyses.

Even in the context of unsupervised learning, where we lack explicit target variables, it is common practice to define and apply a suitable model that captures the relationships and patterns among the data features. This model serves several purposes:

1. It provides a compact representation of the data's underlying structure.
2. It can be used to generate new, synthetic data points that share characteristics with the original dataset.
3. It enables anomaly detection by identifying data points that deviate significantly from the learned patterns.
4. It facilitates interpretation and visualization of high-dimensional data.

The choice of model depends on the specific unsupervised learning task and the nature of the data. For instance:

- For clustering, we might use models like k-means, Gaussian Mixture Models, or hierarchical clustering algorithms.
- For density estimation, kernel density estimation or parametric models such as Gaussian distributions might be employed.
- For dimensionality reduction, techniques like Principal Component Analysis (PCA), t-SNE, or autoencoders could be utilized.

By applying these unsupervised learning techniques and models, we can gain valuable insights into the inherent structure of our data, even in the absence of predefined target variables. This approach is particularly valuable for defining generative models (such as LLM and all generative AI systems), exploratory data analysis, feature engineering, and as a preprocessing step for subsequent supervised learning tasks.

Reinforcement learning

In the paradigm of reinforcement learning, our primary objective is to identify an optimal sequence of actions within a given framework or environment. This sequence is designed to maximize a certain metric, typically referred to as reward or profit.

Reinforcement learning has found success in various domains, such as

- Game Playing: Achieving superhuman performance in chess, Go, and video games
- Robotics: Learning complex motor skills and navigation
- Online Resource Management: Dinamically optimizing allocation of (computational, physical, financial, etc.) resources

Unlike supervised learning, where we have a dataset of labeled examples, reinforcement learning operates in a more dynamic, interactive setting:

Environment Interaction An environment is available which responds to actions taken by an agent. This environment can be represented as a function $E : A \times S \mapsto R \times S$, where:

- A is the set of possible actions
- S is the set of possible states
- R is the set of possible rewards

Reward Mechanism In response to each action $a \in A$ taken by the agent in state $s \in S$, the environment returns: - A reward $r \in \mathbb{R}$, which quantifies the immediate benefit of the action - A new state $s' \in S$, representing the updated condition of the environment

Policy Optimization The goal is to learn a policy $\pi : S \to A$ that maps states to actions in a way that maximizes the expected cumulative reward over time. This can be expressed mathematically as:

$$\pi^* = \underset{\pi}{\text{argmax}}\, \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t | \pi \right]$$

where $\gamma \in [0, 1]$ is a discount factor that balances immediate and future rewards and $r_t|\pi$ is the reward obtained at step t is policy π is applied.

A key challenge in reinforcement learning is balancing **exploration**, that is trying new actions to gather more information about the environment and **exploitation**, leveraging known profitable actions to maximize immediate reward.

Several different approaches exist for solving reinforcement learning problems:

- Model-Based vs. Model-Free, that is Learn a model of the environment's dynamics and use it for planning vs. Learn optimal actions directly without explicitly modeling the environment
- On-Policy vs. Off-Policy, that is learn from actions taken by the current policy being learned vs. learn from actions taken by a different policy (e.g., from stored experiences)

In summary, reinforcement learning provides a framework for solving sequential decision-making problems in complex, uncertain environments. In this framework, it represents a versatile approach for a wide range of real-world dynamic optimization tasks.

Observe that the framework in which reinforcement learning is defined is quite different than the ones of supervised and unsupervised learning, which results in the adoption of quite different mathematical and algorithmic techniques. For this reason, reinforcement learning will not be further considered here.

Formal Definition of a Machine Learning Task

A machine learning task is defined over a pair of domains:

Domain set $\mathcal{X}$ This is the set of objects we wish to label or make predictions about.. Each object $\mathbf{x} \in \mathcal{X}$ is usually modeled as an array of features[1]. The number of features is referred to as the dimensionality of the problem. Formally, we can then represent an object as $\mathbf{x} = [x_1, x_2, ..., x_d]$, where d is the dimensionality.

Label set $\mathcal{Y}$ This is the set of possible label values associated with objects in $\mathcal{X}$. The nature of $\mathcal{Y}$ determines the type of learning task:

- If $\mathcal{Y}$ is continuous, we are dealing with a regression task.
- If $\mathcal{Y}$ is discrete, we have a classification task.
  - When $|\mathcal{Y}| = 2$, we have a case of binary classification.
  - When $|\mathcal{Y}| > 2$, we have multi-class classification.

The learner (an algorithm $\mathcal{A}$) has access to a training set $\mathcal{T}$, a collection of item-label pairs: $\mathcal{T} = \{(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_n, t_n)\}$. We shall usually denote as $\mathbf{X}$ the matrix of items (feature matrix), that is

$$\mathbf{X} = \begin{pmatrix} - & \mathbf{x}_1 & - \\ & \vdots & \\ - & \mathbf{x}_n & - \end{pmatrix}$$

and as $\mathbf{t}$ the vector of labels (target vector), that is

$$\mathbf{t} = \begin{pmatrix} t_1 \\ \vdots \\ t_n \end{pmatrix}$$

The learner is requested to return, for any given training set $\mathcal{T}$, a prediction rule (classifier, regressor) $A = \mathcal{A}(\mathcal{T}) = h : \mathcal{X} \mapsto \mathcal{Y}$. The predictor should be able to generate a prediction y for any item $\mathbf{x} \in \mathcal{X}$: this can be done according to different approaches.

- Direct Target Value Prediction: in this case, the algorithm A predicts a value y which is a guess of the target of $\mathbf{x}$.That is, it directly computes a function $h : \mathcal{X} \mapsto \mathcal{Y}$
- Probability Distribution Prediction: in this case, the algorithm A probability distribution on $\mathcal{Y}$ which, for any $y \in \mathcal{Y}$, A returns an estimated probability $p(y|\mathbf{x})$ that y is the target value of $\mathbf{x}$. In order to return a unique value y, this approach must be accompanied by some independent rule to derive, given $p(y|\mathbf{x})$, the value to be predicted.

Deriving a Functional Predictor

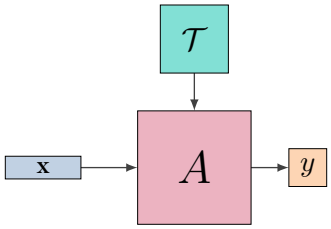
There are multiple approaches to deriving a functional predictor:

Direct Prediction Computation

In this case, a predefined algorithm A is applied for each prediction: the algorithm computes a function $h : \mathcal{X} \times (\mathcal{X} \times \mathcal{Y})^n \mapsto \mathcal{Y}$. In particular, it computes the prediction y for an item $\mathbf{x}$ by computing $h(\mathbf{x}, \mathbf{X}, \mathbf{t})$: that is, it takes into account the entire training set for each prediction.

Observe that no learning (that is, deriving some prediction rule from examples to be applied for predictions) is done.

[1]Actually, in advanced cases objects could have more complex structures, such as for example sequences or graphs.



Example of this approach: k-nearest neighbors algorithm for classification

The class predicted for item **x** is the majority class in the set of k elements of **X** which are nearest to **x** according to a predefined measure

Let's enhance and expand this text, preserving LaTeX commands and providing a more comprehensive overview of this approach to deriving predictors:

Model-Based Learning Approach

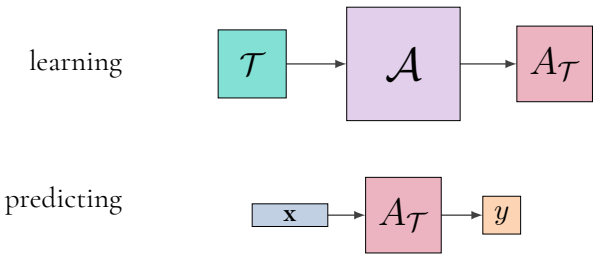
The second approach to deriving a predictor involves learning a function from a predefined class of models. This method, often referred to as the model-based approach, can be formalized as follows:

- Define a class of functions $\mathcal{H} : \mathcal{X} \to \mathcal{Y}$ -
- employ a learning algorithm $\mathcal{A}$ that derives a specific function $h_{\mathcal{T}} \in \mathcal{H}$ from the training set $\mathcal{T}$. That is, $\mathcal{A}$ implements a function from $\mathcal{T}$ to $\mathcal{H}$

The idea, here, is that $\mathcal{A}$ finds the function in $\mathcal{H}$ (indeed, an algorithm $A_{\mathcal{T}}$ implementing this function) that "best" predicts y from **x** when applied to the examples in $\mathcal{T}$, i.e. best predicts t_i from $\mathbf{x}_i$ for all $(\mathbf{x}_i, t_i) \in \mathcal{T}$[2].

For any new item **x**, the corresponding target value is computed as $h_{\mathcal{T}}(\mathbf{x})$, that is by applying $A_{\mathcal{T}}$ on input **x**.

As we will see, a relevant case here is when $\mathcal{H}$ is a set of parametric functions (denote it as $\mathcal{H}_{\boldsymbol{\theta}}$, where $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)$ is the set of parameters) with the same structure and which differs each other by the values of the parameters in $\boldsymbol{\theta}$. In this case, searching a function in $\mathcal{H}_{\boldsymbol{\theta}}$ is equivalent to searching a value for $(\theta_1, \ldots, \theta_m)$.



A simple example of this approach is linear regression, where the value predicted for item **x** is computed as the linear combination of its feature values $x_1, x_2, \ldots, x_d$, each weighted by a suitable constant parameter $w_1, w_2, \ldots, w_d$, plus a bias w_0. That is, the prediction is computed as

$$y = \sum_{i=1}^{d} w_i x_i + w_0 = \mathbf{w}^T \bar{\mathbf{x}}$$

where **w** is the vector[3] $\begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix}$ and $\bar{\mathbf{x}}$ is the vector $\begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix}$.

[2]Observe, that we need to specify what "best" means here, that is which measure of prediction quality we apply.

[3]In general, all vectors introduced in the following will be assumed as column vectors.

Observe that, in this case, the set of functions $\mathcal{H}$ is the set of $d+1$-dimensional linear functions, parameterized by $w_0, \ldots, w_d$.

The $d+1$ values $w_0, w_1, \ldots, w_d$ are learned by $\mathcal{A}$ from the training set $\mathcal{T}$.

Ensemble Learning Approach

The third approach involves creating a whole collection of predictors and perform predictions by composing the set of their predictions, which can often lead to improved performance and robustness. This method can be better specified as follows:

- Ensemble Construction: derive from the training set $\mathcal{T}$ a set of s algorithms $A^{(1)}_{\mathcal{T}}, \ldots, A^{(s)}_{\mathcal{T}}$, each computing a different function $h^{(i)}_{\mathcal{T}} : \mathcal{X} \to \mathcal{Y}$ in the given class $\mathcal{H}$. Each $A^{(i)}$ is a then predictor of y from $\mathbf{x}$ derived from the same set $\mathcal{T}$ of examples.
- Weight Assignment: assign a set of corresponding weights $w^{(1)}, \ldots, w^{(s)}$ to each predictor. Essentially, each weight $w^{(i)}$ is directly related to estimated quality of the predictions provided by $A^{(i)}$ for what concerns items in $\mathcal{T}$.
- Prediction Combination: for any $\mathbf{x}$, compute the final predicted value by combining the values $y^{(1)} = h^{(1)}_{\mathcal{T}}(\mathbf{x}), \ldots, y^{(s)} = h^{(s)}_{\mathcal{T}}(\mathbf{x})$ predicted by the individual algorithms, weighted by their respective weights $w^{(1)}, \ldots, w^{(s)}$.

Formal Definition:

The target value predicted for item $\mathbf{x}$ is the linear combination of the values $y^{(1)}, y^{(2)}, \ldots, y^{(s)}$, predicted by predictors $A^{(1)}, A^{(2)}, , \ldots, A^{(s)}$, each weighted by the corresponding weight $w^{(1)}, w^{(2)}, \ldots, w^{(s)}$.

Each $A^{(i)}$ is a simple predictor derived from $\mathcal{T}$

An important variant of this approach is represented by fully bayesian prediction, where the set of different predictors is a continuous one, each corresponding to a different value of a set of parameters $(w_1, \ldots, w_d) \in \mathbb{R}^d$. In this case, clearly, the sum is substituted by a (usually multidimensional) integral

Examples of the third approach: for regression tasks, learn predictors as linear regressors and compute the final prediction as for item $\mathbf{x}$ is the linear combination of the values $y^{(1)}, y^{(2)}, \ldots, y^{(s)}$, predicted by predictors $A^{(1)}, A^{(2)}, , \ldots, A^{(s)}$, each weighted by the corresponding weight $w^{(1)}, w^{(2)}, \ldots, w^{(s)}$.

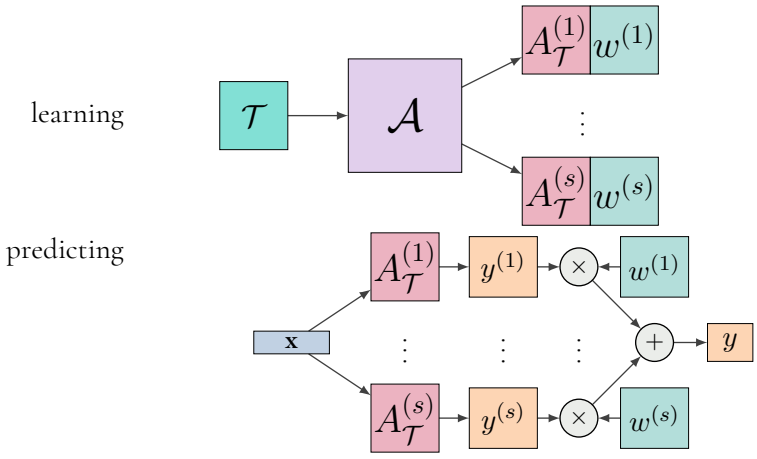
$$h(\mathbf{x}) = \sum_{i=1}^{s} w^{(i)} h^{(i)}(\mathbf{x})$$

where $\sum_{i=1}^{s} w^{(i)} = 1$ and $w^{(i)} \geq 0$ for all i.

For classification tasks, we can use a voting approach, where the class predicted by the largest number of predictors is returned

$$h(\mathbf{x}) = \operatorname*{argmax}_{y \in \mathcal{Y}} \sum_{i=1}^{s} w^{(i)} \mathbb{1}[h^{(i)}(\mathbf{x}) = y]$$

where $\mathbb{1}[\cdot]$ is the indicator function, which is one if the predicate in square bracket is true and 0 otherwise.



An important variant of this approach is represented by fully bayesian prediction, where the set of different predictors is a continuous one, each corresponding to a different value of the set of parameters $(w_1, \ldots, w_d) \in \mathbb{R}^d$. In this case, clearly, the sum is substituted by a (usually multidimensional) integral

The three approaches differ since:

- in the first case, a predefined algorithm is applied to input data comprising both the item $\mathbf{x}$ and the whole training set $\mathcal{T}$
- in the second case, an algorithm to be applied to any item $\mathbf{x}$ is derived in dependance from the training set $\mathcal{T}$
- in the third case, no single algorithm is applied to $\mathbf{x}$; the prediction is instead computed from the predictions returned by a set of predictors

Probabilistic Framework for Machine Learning

Training Object Generation Model: We posit that objects in the training set are sampled from $\mathcal{X}$ according to an unknown (marginal) probability distribution p_M. Formally, for any $\mathbf{x} \in \mathcal{X}$, $p_M(\mathbf{x})$ represents the probability that $\mathbf{x}$ is the next object sampled in the training set.

Training Target Generation Model: In the general case, we assume that labels associated with items in the training set are generated according to a probability distribution p_C, conditional on $\mathcal{X}$. Specifically, for any $t \in \mathcal{Y}$, $p_C(t|\mathbf{x})$ denotes the probability that the observed label of object $\mathbf{x}$ in the training set is t: this is equivalent to assuming a joint distribution $p(\mathbf{x}, t) = p_M(\mathbf{x})p_C(t|\mathbf{x})$ for the generation of item-target pairs. For simplicity, we initially assume a deterministic relationship between objects and labels, represented by an unknown function f such that $t = f(\mathbf{x})$.

Focusing on the model-based approach described earlier, several key concepts emerge:

- The quality of a predictor h, such as one returned by the learner, is evaluated in terms of risk. For any element $\mathbf{x} \in \mathcal{X}$, the error of h when applied to $\mathbf{x}$ stems from the comparison of its prediction $h(\mathbf{x})$ and the correct target label t. This comparison is quantified using a predefined loss function $L : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$.
- The error of a prediction $y = h(\mathbf{x})$ is then defined in terms of prediction risk, given by applying the loss:

$$\mathcal{R}_p(y, \mathbf{x}) \triangleq L(y, f(\mathbf{x}))$$

- In the more general case, where only a probabilistic relation $p_C(y|\mathbf{x})$ (instead of a function f) is assumed between an item and its corresponding label, the prediction risk corresponds to the expectation of the loss function with respect to this distribution:

$$\mathcal{R}_{p_C}(y, \mathbf{x}) \triangleq \mathop{\mathbb{E}}_{t \sim p_C(\cdot|\mathbf{x})} [\, L(y, t) \,] = \int_{\mathcal{Y}} L(y, t) \cdot p_C(t|\mathbf{x}) dt$$

- For classification tasks, this becomes:

$$\mathcal{R}_{p_C}(y, \mathbf{x}) \triangleq \mathop{\mathbb{E}}_{t \sim p_C(\cdot|\mathbf{x})} [\, L(y, t) \,] = \sum_{t \in \mathcal{Y}} L(y, t) \cdot p_C(y|\mathbf{x})$$

In the following, we will sometimes refer to the simplest case of a binary classification task, where $\mathcal{Y} = \{0, 1\}$, with the 0-1 loss function

$$L(y, t) = \mathbb{1}[y \neq t],$$

which returns 1 if the arguments differ and 0 otherwise.

This probabilistic framework provides a rigorous foundation for understanding and evaluating machine learning models, accounting for both the inherent uncertainty in data generation and the performance of predictive algorithms.

Expected and Empirical Risk

The error of a predictor h is defined in terms of expected loss over all objects in $\mathcal{X}$:

$$\mathcal{R}_{p_M, f}(h) \triangleq \mathop{\mathbb{E}}_{\mathbf{x} \sim p_M} [\, L(h(\mathbf{x}), f(\mathbf{x})) \,] = \int_{\mathcal{X}} L(h(\mathbf{x}), f(\mathbf{x})) \cdot p_M(\mathbf{x}) d\mathbf{x}$$

and in the general case:

$$\mathcal{R}_p(h) = \mathop{\mathbb{E}}_{(\mathbf{x}, y) \sim p} [\, L(h(\mathbf{x}), t) \,] = \int_{\mathcal{X}} \int_{\mathcal{Y}} L(h(\mathbf{x}), t) \cdot p_M(\mathbf{x}) \cdot p_C(t|\mathbf{x}) d\mathbf{x} dt$$

In the case of 0-1 loss, this is just the probability of wrong prediction in a randomly sampled item or pair, repectively, that is

$$\mathcal{R}_{p_M, f}(h) = \mathop{\mathbb{E}}_{\mathbf{x} \sim p_M} [\, \mathbb{1}[h(\mathbf{x}) \neq f(\mathbf{x})] \,] = \mathop{\mathbb{P}}_{\mathbf{x} \sim p_M} [\, h(\mathbf{x}) \neq f(\mathbf{x}) \,]$$

or

$$\mathcal{R}_p(h) = \mathop{\mathbb{E}}_{(\mathbf{x}, t) \sim p} [\, \mathbb{1}[h(\mathbf{x}) \neq t] \,] = \mathop{\mathbb{P}}_{(\mathbf{x}, t) \sim p} [\, h(\mathbf{x}) \neq t \,]$$

Since p_M and f (or p) are unknown, the risk can only be estimated from the available data (the training set $\mathcal{T}$). This leads to the definition of **empirical risk** $\overline{\mathcal{R}}_{\mathcal{T}}(h)$, which provides an estimate of the expectation of the loss function as the average loss on the training set:

$$\overline{\mathcal{R}}_{\mathcal{T}}(h) = \frac{1}{|\mathcal{T}|} \sum_{(x, t) \in \mathcal{T}} L(h(\mathbf{x}), t)$$

In the case of 0-1 loss, this is the fraction of elements of $\mathcal{T}$ which are misclassified by h

$$\overline{\mathcal{R}}_{\mathcal{T}}(h) = \frac{1}{|\mathcal{T}|} \sum_{(\mathbf{x}, t) \in \mathcal{T}} \mathbb{1}[h(\mathbf{x}) \neq t] = \frac{|\{(\mathbf{x}, t) \in \mathcal{T} \mid h(\mathbf{x}) \neq t\}|}{|\mathcal{T}|}$$

In this way, a learning problem is reduced to a minimization problem in some functional space $\mathcal{H}$, the set of all possible predictors h.

$$h_{\mathcal{T}} = \underset{h \in \mathcal{H}}{\operatorname{argmin}}\, \overline{\mathcal{R}}_{\mathcal{T}}(h)$$

Here, $\mathcal{H}$ is the set of **hypotheses** or **inductive bias**

Issues related to the inductive bias

The choice of the set of hypotheses is an important issue in ML. In particular, we may ask

- what is the effect of the structure and size of $\mathcal{H}$?
- how to define $\mathcal{H}$ in such a way to make it feasible to compute $h_{\mathcal{T}}$?

For what concerns the choice of the hypotheses class $\mathcal{H}$, it can be viewed as reflecting some prior knowledge that the learner has about the task, in terms of a belief that one of the members of the class $\mathcal{H}$ is a low-error predictor for the task.

A trivial way of pursuing the goal of deriving predictors with minimal risk would be to define a very rich class, that is assuming that many possible functions belong to $\mathcal{H}$: as a limit, $\mathcal{H}$ could be defined just as the set of all functions $f : \mathcal{X} \mapsto \mathcal{Y}$.

This approach, however, can be easily seen to induce problems.

Assume, in fact, a binary classification problem with training set $\mathcal{T} = (\mathbf{X}, \mathbf{t})$, with $0/1$ loss

$$L(y, t) = \begin{cases} 0 & \text{if } y = t \\ 1 & \text{otherwise} \end{cases}$$

that is, the loss is 1 if the item is misclassified, 0 otherwise. As a consequence, the risk is the expected number of classification errors, while the empirical risk is the fraction of items in the training set which are misclassified.

Assume also that $p_C(t = 1|\mathbf{x}) = \frac{1}{2}$ for $\mathbf{x} \in \mathcal{X}$, that is, the two classes have same size in the population.

If we consider the classification function defined as:

$$h_{\mathcal{T}}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} = \mathbf{x}_i \in \mathbf{X}, t_i = 1 \\ 0 & \text{otherwise} \end{cases}$$

that is, a predictor that assigns to class 1 all items labeled as 1 in the training set, while all other items are classified as 0.

Clearly, the empirical risk here is 0 by definition, but the risk is $\approx \frac{1}{2}$. When applied to a dataset randomly sampled from the population, the quality of $h_{\mathcal{T}}$ is the same of a function which randomly assigns items to classes.

This is called **overfitting**: the classification method behaves well on the training set, but poorly on other items from the population.

However, if $\mathcal{H}$ is very small, it may happen that no predictor from this set is able to provide an acceptably small risk.

Reassuming, the following general considerations can be done for what concerns the size of $\mathcal{H}$.

- If $\mathcal{H}$ is too large (complex), **overfitting** may occur: a function which behaves very well on the training set may be available which however performs poorly on new data
- If $\mathcal{H}$ is too small (simple), **underfitting** may occur: no function behaving in a satisfactory way, both on the training set and on new sets of data, is available in $\mathcal{H}$

This is related to the so-called **bias variance tradeoff**.