Machine learning

Foundations

Corso di Laurea Magistrale in Informatica

Università di Roma Tor Vergata

Giorgio Gambosi

a.a. 2024–2025

Cornerstone of machine learning

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.
Fundamental assumption: these data, far from being random, possess an intrinsic structure—albeit one that often proves elusive to identify.

Unsupervised Learning

We want to extract insights into the underlying structure of the data, deepening our understanding of its inherent nature.

- involves techniques such as clustering, dimensionality reduction, or generative modeling.
- particularly compelling application of unsupervised learning: 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,

Supervised Learning

We want 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.

n both these scenarios, we start with a set of observations already produced by the unknown process.

Reinforcement learning

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.

Common scenario 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.
- In the case of supervised learning, the training set is augmented with 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 $\mathbf{x}_i$.

Representations

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.

Supervised learning

Task: 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 $t \in \{1, \ldots, K\}$

Supervised learning

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

Supervised learning

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.

Unsupervised learning

The objective here is to detect inherent patterns and structures within a given collection of items, known as the dataset $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, 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.

Unsupervised learning

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.

Unsupervised learning

By applying 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

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. 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:

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

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.

Reinforcement learning

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.

Supervised learning framework: domains

Domain set $\mathcal{X}$: Set of objects we may wish to label. Each object is modeled as a vector of features. The number of features is the dimensionality of the problem

Label set $\mathcal{Y}$: Set of possible label values associated to objects in $\mathcal{X}$.

- $\mathcal{Y}$ continuous: regression
- $\mathcal{Y}$ discrete: classification

Supervised learning framework: input data

Training set $\mathcal{T}$: A set of object-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 objects (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

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

Supervised learning framework: deriving a predictor

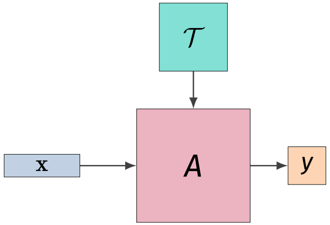
- A predictor algorithm A must be derived from $\mathcal{T}$, which returns a prediction y for any item $\mathbf{x} \in \mathcal{X}$
- This can be done according to different approaches.
- This depends from what is the “prediction” we wish to obtain:
  1. the prediction is a target value: in this case, A predicts a value y which is a guess of the target of $\mathbf{x}$. That is, it computes a function $h : \mathcal{X} \mapsto \mathcal{Y}$
  2. the prediction is a probability distribution on $\mathcal{Y}$: in this case, A returns, for any $y \in \mathcal{Y}$, an estimate probability $p(y|\mathbf{x})$ that y is the target value of $\mathbf{x}$

Supervised learning framework: deriving a functional predictor

First approach: apply a given algorithm A computing a function $h : \mathcal{X} \times (\mathcal{X} \times \mathcal{Y})^n \mapsto \mathcal{Y}$

- A predicts y from $\mathbf{x}$ by computing $h(\mathbf{x}, \mathbf{X}, \mathbf{t})$

Supervised learning framework: deriving a functional predictor


$$\begin{tikzcd} & \mathcal{T} \arrow[d] & \\ \mathbf{x} \arrow[r] & A \arrow[r] & y \end{tikzcd}$$

Supervised learning framework: deriving a functional predictor

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

The class predicted for item $\mathbf{x}$ is the majority class in the set of k elements of $\mathbf{X}$ which are nearest to $\mathbf{x}$ according to a predefined measure

Supervised learning framework: deriving a functional predictor

Second approach: derive from $\mathcal{T}$ an algorithm $A_{\mathcal{T}}$ computing a function $h_{\mathcal{T}} : \mathcal{X} \mapsto \mathcal{Y}$ in a given class

- A is the algorithm in a predefined class which "best" predicts y from $\mathbf{x}$ when applied to the set of examples in $\mathcal{T}$
- this can be done by means of a learning algorithm $\mathcal{A}$ which derives A from $\mathcal{T}$
- $A_{\mathcal{T}} = h : \mathcal{X} \mapsto \mathcal{Y}$

Supervised learning framework: deriving a functional predictor

learning

$$\begin{tikzcd} \mathcal{T} \arrow[r] & \mathcal{A} \arrow[r] & A_{\mathcal{T}} \end{tikzcd}$$

predicting

$$\begin{tikzcd} \mathbf{x} \arrow[r] & A_{\mathcal{T}} \arrow[r] & y \end{tikzcd}$$

Supervised learning framework: deriving a functional predictor

Example of second approach: linear regression

The target value predicted for item $\mathbf{x}$ is the linear combination of its feature values $x_1, x_2, \ldots, x_d$, each weighted by a suitable value $w_1, w_2, \ldots, w_d$, plus a bias value w_0. That is,

$$y = \sum_{i=1}^{d} w_i x_i + w_0$$

The $d+1$ values $w_0, w_1, \ldots, w_d$ are learned in dependance of the training set $\mathcal{T}$.

Supervised learning framework: deriving a functional predictor

Third approach: derive from $\mathcal{T}$ a set of algorithms $A_{\mathcal{T}}^{(1)}, \ldots, A_{\mathcal{T}}^{(s)}$ each computing a different function $h_{\mathcal{T}}^{(i)} : \mathcal{X} \mapsto \mathcal{Y}$ in a given class, and a set of corresponding weights $w^{(1)}, \ldots, w^{(s)}$.

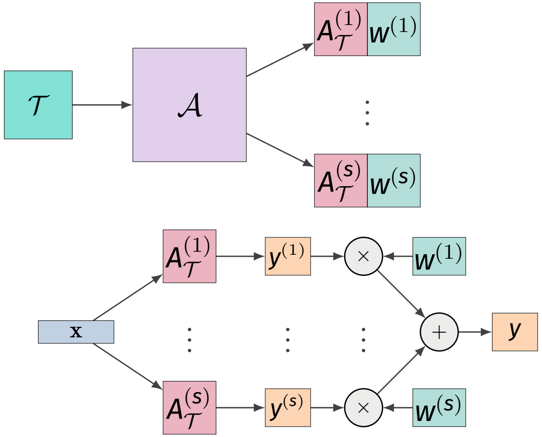
Compute the predicted value combining the values $y^{(1)}, \ldots, y^{(s)}$ predicted by the algorithms, weighted by the weights $w^{(1)}, \ldots, w^{(s)}$

- each $A_{\mathcal{T}}^{(i)}$ is a predictor of y from $\mathbf{x}$ derived from the set of examples in $\mathcal{T}$
- the estimated quality of the predictions provided by $A_{\mathcal{T}}^{(i)}$ is represented by the weight $w^{(i)}$

Supervised learning framework: deriving a functional predictor

learning

predicting



Supervised learning framework: deriving a functional predictor

Example of third approach: ensemble methods

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

Supervised learning framework: deriving a functional predictor

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 $\mathbf{X}, \mathbf{t}$
- in the second case, an algorithm to be applied to any item $\mathbf{x}$ is derived in dependance from the training set $\mathbf{X}, \mathbf{t}$
- 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

Supervised learning framework: learning framework

Training objects generation model: Items in the training set are assumed sampled from $\mathcal{X}$ according to a probability distribution p_M. That is, for any $\mathbf{x} \in \mathcal{X}$, $p_M(\mathbf{x})$ is the probability that $\mathbf{x}$ is the next item sampled in the training set

Training targets generation model: In the general case, we assume the labels associated to the items in the training set are generated according to a probability distribution p_C conditional on $\mathcal{X}$. That is, for any $t \in \mathcal{Y}$, $p_C(t|\mathbf{x})$ is the probability that the observed label of object $\mathbf{x}$ in the training set is t. For the moment, we shall assume that the relation between object and label is deterministic, that is there exists an unknown function f such that $t = f(\mathbf{x})$

Supervised learning framework: prediction risk

Let us restrict ourselves, in the following, to the second approach described above. Then, some concepts are relevant. Given any element $\mathbf{x} \in \mathcal{X}$:

Error: The error of a predictor h derives from the comparison of its prediction $h(\mathbf{x})$ and the correct target label t.

Loss: The comparison is performed by applying a predefined loss function $L : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$.

Risk of prediction: The error of a prediction y is defined in terms of prediction risk as given by applying the loss

$$\mathcal{R}_f(y, \mathbf{x}) = L(y, f(\mathbf{x}))$$

In the general case when only a probabilistic relation $p_C(t|\mathbf{x})$ is assumed between label and target, this corresponds to the loss expection given the prediction y, with the target t varying according to $p_C(t|\mathbf{x})$

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

or, in the case of classification

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

Bayes estimator

In this framework, the optimal prediction is the one which minimizes the risk,

$$h^*(\mathbf{x}) = \underset{y}{\mathrm{argmin}}\, \mathcal{R}_{p_C}(y, \mathbf{x}) = \underset{y}{\mathrm{argmin}}\, E_{p_C}[L(y,t)]$$

that is,

$$h^*(\mathbf{x}) = \underset{y}{\mathrm{argmin}}\, \mathcal{R}_f(y, \mathbf{x}) = \underset{y}{\mathrm{argmin}}\, L(y, f(\mathbf{x}))$$ in the simpler case

$$h^*(\mathbf{x}) = \underset{y}{\mathrm{argmin}}\, \mathcal{R}_{p_C}(y, \mathbf{x}) = \underset{y}{\mathrm{argmin}} \int_{\mathcal{Y}} L(y,t) \cdot p_C(t|\mathbf{x}) dt$$ in the general case

in the general case, this is denoted as Bayes estimator.
Unfortunately, this approach cannot be applied since both the function f and the distribution p_C are assumed unknown.

Machine learning framework: predictor risk

The error of a predictor h is defined in terms of risk expected loss on all items in $\mathcal{X}$

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

In the general case,

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

Machine learning framework: learner evaluation

Since p_M and p_C (or f) are not known, the risk can only be estimated from the data available (the training set $\mathcal{T}$).

Empirical risk: The risk can be estimated from the training set by estimating the expectation of the loss function as the average loss on the set.

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

Machine learning framework: from learning to optimization

The fundamental approach in the approach to machine learning considered here is deriving a predictor h which (at least approximately) minimizes the empirical risk computed on the available training set.

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

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

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 Machine Learning:

- 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^*?

Choice of the set of hypotheses

- The hypotheses class $\mathcal{H}$ can be viewed as reflecting some prior knowledge that the learner has about the task
  - a belief that one of the members of the class $\mathcal{H}$ is a low-error model for the task
- A trivial way of pursuing this goal 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 $f : \mathcal{X} \mapsto \mathcal{Y}$

Choice of the set of hypotheses

Problem with large $\mathcal{H}$:

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

Choice of the set of hypotheses

Consider the classification function defined as:

$$h(x) = \begin{cases} 1 & \text{if } \mathbf{x} = \mathbf{x}_i \in \mathbf{X} \\ 0 & \text{otherwise} \end{cases}$$

that is, h assigns to class 1 all items labeled as 1 in the training set. 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^* 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 new data from the population.

Choice of the set of hypotheses

With respect to $\mathcal{H}$, the following considerations can be done:

- 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