Machine learning

Foundations

Corso di Laurea Magistrale in Informatica

Università di Roma Tor Vergata

Giorgio Gambosi

a.a. 2022–2023

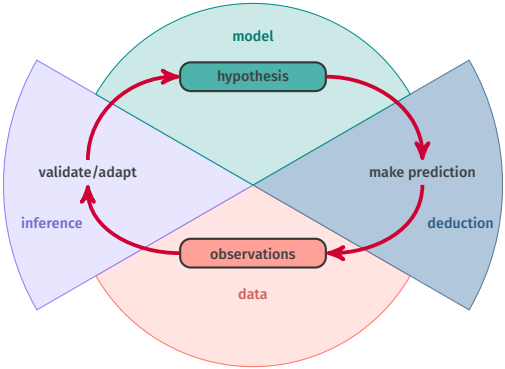
Objectives

Machine learning: inductive approach

Learning of commonalities through analysis of a set of examples (training set), which is assumed to be available.

- A training set of n items is represented as a set of input vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$, used to derive a model.
- If the purpose is item classification with respect to a collection of predefined classes, the training set also includes a target vector $\mathbf{t} = \{t_1, \ldots, t_n\}$, where the class of each training set item is specified.

The learning process



Types of problems

Supervised learning

- Predict, given the values of a set of characteristics (features) of an item $\mathbf{x}$, the unknown value of an additional characteristic (target) of the item
  - Target in $\mathbb{R}$: regression. Target in $\{1, \ldots, K\}$: classification.
- General approach: define (by means of learning from a set of examples) a predictor of the target value from the set of feature values.
- The training set $\mathcal{T} = (\mathbf{X}, \mathbf{t})$ provides a set of examples of the relation between set of features and target: each example includes a feature vector $\mathbf{x}_i = \{x_{i1}, \ldots, x_{im}\}$ and the corresponding target t_i.
- The predictor could be:
  1. a function $y()$ which, for any item $\mathbf{x}$, returns a value $y(\mathbf{x})$ as an estimate of t
  2. a probability distribution which associates to each possible value $\overline{y}$ in the target domain, the corresponding probability $p(y = \overline{y}|\mathbf{x})$

Types of problems

Unsupervised learning

- We wish to extract, from a given collection of items (dataset) $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, with no target associated, some synthetic information, such as:
  - subsets of similar items (clustering)
  - the distribution of items in their domain (density estimation)
  - the projection, as informative as possible, of items on lower dimensional subspaces, that is, their characterization by means of a smaller set of features (feature selection, feature extraction)
- This is often performed by deriving a suitable model, of the data features.

Reinforcement learning

- We want to identify, in a given framework, a sequence of actions to be performed in order to maximize a certain profit
- As in supervised learning, no examples are given, but an environment is available which returns a profit in correspondance to the execution of any action

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: $\mathcal{T} = \{(\mathbf{x}_1, t_1), \dots, (\mathbf{x}_n, t_n)\}$. 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

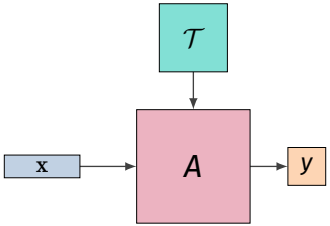
- A predictor algorithm A must be derived from $\mathcal{T}$, which returns a prediction y for any item $\mathbf{x} \in \mathcal{X}$
- This can be done according to different approaches.
- This depends from what is the "prediction" we wish to obtain:
  1. the prediction is a target value: in this case, 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



Supervised learning framework: deriving a functional predictor

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

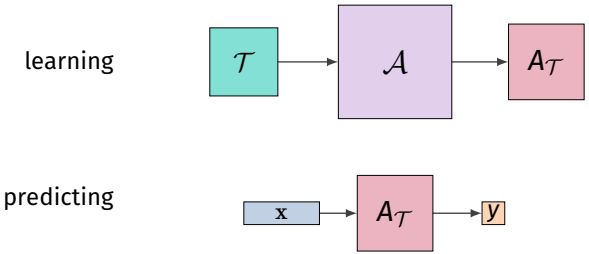
The class predicted for item $\mathbf{x}$ is the majority class in the set of *k* elements of $\mathbf{X}$ which are nearest to $\mathbf{x}$ according to a predefined measure

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



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 $w^{(1)}, \ldots, w^{(s)}$.

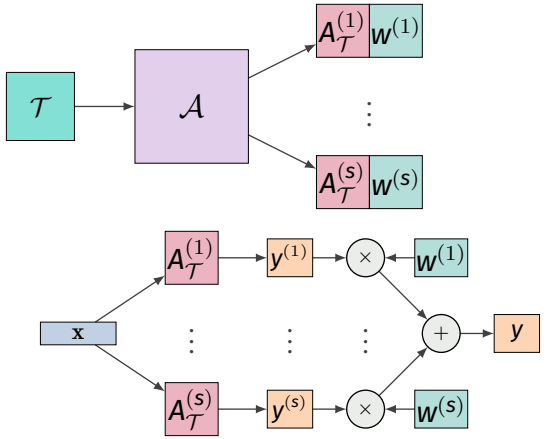
Compute the predicted value combining the values $y^{(1)}, \ldots, y^{(s)}$ predicted by the algorithms, weighted by the weights $w^{(1)}, \ldots, w^{(s)}$

- 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_1. That is, for any $\mathbf{x} \in \mathcal{X}$, $p_1(\mathbf{x})$ 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_2 conditional on $\mathcal{X}$. That is, for any $t \in \mathcal{Y}$, $p_2(t|\mathbf{x})$ 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 $\hat{y}$ is defined in terms of prediction risk as given by applying the loss

$$\mathcal{R}(\hat{y}, t) = L(h(\mathbf{x}), t)$$

In the general case when only a probabilistic relation $p_2(t|\mathbf{x})$ is assumed between label and target, this corresponds to

$$\mathcal{R}(\hat{y}, \mathbf{x}) = E_{p_2}[L(\hat{y}, t)] = \int_{\mathcal{Y}} L(\hat{y}, t) \cdot p_2(t|\mathbf{x}) dt$$

or, in the case of classification

$$\mathcal{R}(\hat{y}, \mathbf{x}) = E_{p_2}[L(\hat{y}, t)] = \sum_{t \in \mathcal{Y}} L(\hat{y}, t) \cdot p_2(t|\mathbf{x})$$

Bayes estimator

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

$$y^*(\mathbf{x}) = \underset{\hat{y}}{\operatorname{argmin}}\, \mathcal{R}(\hat{y}, \mathbf{x}) = \underset{\hat{y}}{\operatorname{argmin}}\, E_{p_2}[L(\hat{y}, t)]$$

that is,

$$y^*(\mathbf{x}) = \underset{\hat{y}}{\operatorname{argmin}}\, L(\hat{y}, f(\mathbf{x})) \qquad \text{in the simpler case}$$

$$y^*(\mathbf{x}) = \underset{\hat{y}}{\operatorname{argmin}}\, E_{p_2}[L(\hat{y}, t)] = \underset{\hat{y}}{\operatorname{argmin}} \int_{\mathcal{Y}} L(\hat{y}, t) \cdot p_2(t|\mathbf{x}) dt \qquad \text{in the general case}$$

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_2 of $p(t|\mathbf{x})$ 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 $\mathcal{X}$

$$\mathcal{R}(h) = E_{p_1,f}[L(h(\mathbf{x}), f(\mathbf{x}))] = \int_{\mathcal{X}} L(h(\mathbf{x}), f(\mathbf{x})) \cdot p_1(\mathbf{x}) d\mathbf{x}$$

In the general case,

$$\mathcal{R}(h) = E_{p_1,p_2}[L(h(\mathbf{x}), t)] = \int_{\mathcal{X}} \int_{\mathcal{Y}} L(h(\mathbf{x}), t) \cdot p_1(\mathbf{x}) \cdot p_2(t|\mathbf{x}) d\mathbf{x} dt$$

Machine learning framework: learner evaluation

Since p_1 and p_2 (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.

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

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

Assuming a finite set of hypotheses

The simplest type of restriction on $\mathcal{H}$ is assuming that the number of possible predictors is upper bounded.

In this case, choosing $h_{\mathcal{T}}$ will not overfit, provided $\mathcal{T}$ is sufficiently large

We consider here the task of binary classification with $0-1$ loss function.

Realizability assumption

Assume there exists a predictor in $h^* \mathcal{H}$ which does not make any error in classifying items in $\mathcal{X}$, that is such that

$$\mathcal{R}(h^*) = E_{p_1,f}[L(h^*(\mathbf{x}), f(\mathbf{x}))] = 0$$

This is denoted as realizability assumption

It implies that for any training set $\mathcal{T}$ it results

$$\overline{\mathcal{R}}_{\mathcal{T}}(h^*) = \frac{1}{|\mathcal{T}|} \sum_{(x,t)\in\mathcal{T}} L(h^*(x), t) = 0$$

that is, h^* correctly predicts all items in $\mathcal{T}$

Realizability assumption

$h_{\mathcal{T}}$ correctly classifies all items in $\mathcal{T}$, since otherwise a better predictor would exist but, by hypothesis, $h_{\mathcal{T}}$ is a best predictor for $\mathcal{T}$, even if not necessarily h^* itself

$h_{\mathcal{T}}$ correctly classifies all items in $\mathbf{X}$ as h^* does, but h^* also correctly classifies all other items in $\mathcal{X}$, while this may not hold for $h_{\mathcal{T}}$

while $\mathcal{R}(h^*) = 0$, we only know that $\overline{\mathcal{R}}_{\mathcal{T}}(h_{\mathcal{T}}) = 0$, while $\mathcal{R}(h_{\mathcal{T}})$ is unknown

Realizability assumption

$\mathcal{R}(h_{\mathcal{T}})$ depends on the training set $\mathcal{T}$, which is a sample from the population.

Consider the probability δ of sampling a training set $\mathcal{T}$ which is not representative of population. That is, $\mathcal{R}(h_{\mathcal{T}})$ is "too large", for example larger than a given threshold ε

Realizability assumption

For any given $n > 0$ the probability of sampling a training set of size n that misclassifies an item with probability at least ε is upper bounded by

$$|\mathcal{H}|e^{-\varepsilon n}$$

As a consequence, in order to have a probability δ of selecting a training set which results into a predictor misclassiying an item with probability at most ε, the size of the training set must be

$$n \geq \frac{1}{\varepsilon} \log \frac{|\mathcal{H}|}{\delta}$$

Bias vs variance

The risk associated to the h^*, the predictor which minimizes the empirical risk, can be decomposed in two parts:

$$\mathcal{R}(h^*) = \epsilon_B + \epsilon_V$$

where:

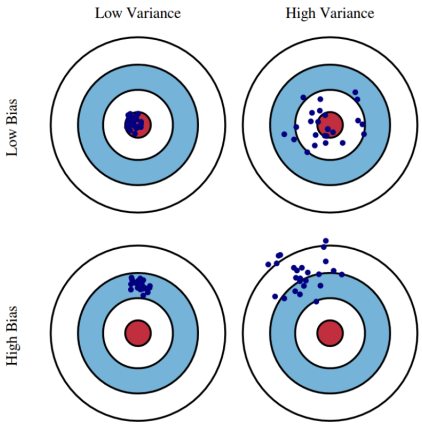
- ϵ_B is the minimum risk achievable by any $h \in \mathcal{H}$: this is only determined by the inductive bias, and independent from the training set. It is a property of the class of hypotheses considered with respect to the prediction task. This is called bias
- ϵ_V is the difference between the above minimum risk in $\mathcal{H}$ and the risk associated to the best predictor in $\mathcal{H}$ with respect to the training set: it is related to the fact that empirical risk minimization only provides an estimate of the best predictor achievable for the given inductive bias. It is a measure of how well the predictor computed from a particular training set approximates the best possible one. Its expectation with respect to all possible training sets is a measure of how much a predictor derived from a random training set may result in poorer performances with respect to the best possible one. This is called variance

Bias vs variance

The choice of $\mathcal{H}$ is subject to a bias-variance tradeoff: higher bias tend to induce lower variance, and vice versa.

- High bias and low variance implies that all predictors which can be obtained from different training sets tend to behave similarly, with a similar risk (low variance). However, all of them then to behave poorly (high bias), since $\mathcal{H}$ is too poor to include a satisfactory predictor for the task considered. This results into underfitting
- Low bias and high variance implies that lot of predictors are available in $\mathcal{H}$, and among them a good one is usually avaliable (low bias). However, quite different predictors can be obtained from different training sets, which implies that it may easily happen that, while a very good performance can be obtained on the training set, the resulting predictor can behave quite differently and more poorly that the best possible one, which implies overfitting

Bias vs variance



Bias vs variance