Foundations

Giorgio Gambosi

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.

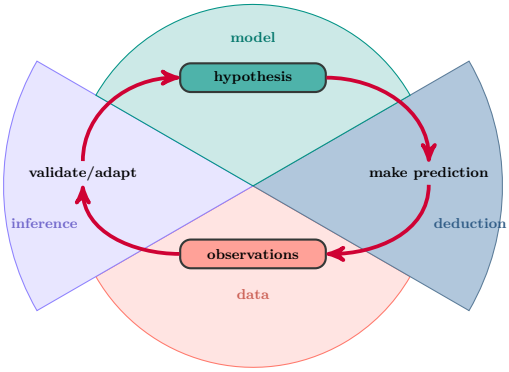


Figure 1: The learning process

Supervised learning

- We want to predict, given the values of a set (features) of an item $\mathbf{x}$, the unknown value of an additional feature target of the item
  - Target in $\mathbb{R}$: regression. Target in $\{1, \ldots, K\}$: classification.
- General approach: defined (by means of learning from a set of examples) a model of the relation between feature and target 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 model 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 $p(y = \overline{y}|\mathbf{x})$

Unsupervised learning

- We wish to extract, from a given collection of items dataset) $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, 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)
- A suitable model, of just the data features, is usually defined and applied also in the case of unsupervised learning.

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

1 Machine learning framework

A task is defined over a pair of domains:

Domain set $\mathcal{X}$: The set of objects we may wish to label. Each object is usually modeled as an array 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 $\mathcal{X}$. If $\mathcal{Y}$ is continuous, we are dealing with a regression task. If it is discrete, a classification task is defined. Assume for the moment that $|\mathcal{Y}| = 2$, this is case of binary classification

The learner (an algorithm $\mathcal{A}$) has access to a training set $\mathcal{T}$, a collection 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

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

The learner is requested to return, for a given training set $\mathcal{T}$, a prediction rule (classifier, regressor) $A(\mathcal{T}) = h : \mathcal{X} \mapsto \mathcal{Y}$

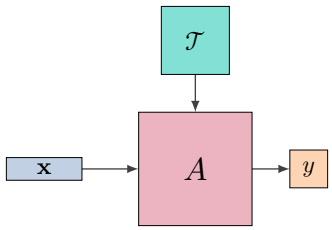
*Actually, in advanced cases objects could have more complex structures, such as for example sequences or graphs.

- 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 $\mathbf{x}$

1.1 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 $h(\mathbf{x}, \mathbf{X}, \mathbf{t})$

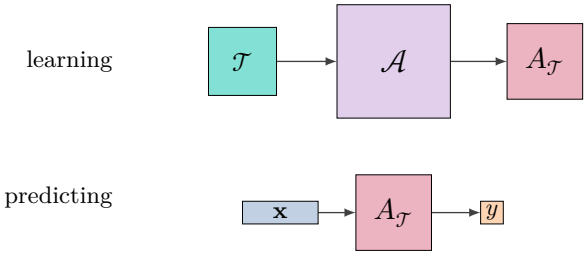


Example of first approach: k-nearest neighbors algorithm for classification

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

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 $\mathcal{T}$
- $A_{\mathcal{T}} = h : \mathcal{X} \mapsto \mathcal{Y}$



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,

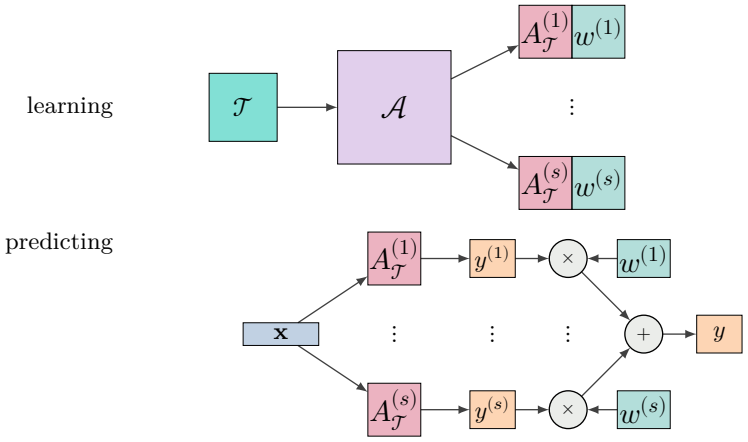
$$y = \sum_{i=1}^{d} w_i x_i + w_0$$

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

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 $w^{(i)}$



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

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

1.2 Modeling assumptions

Training objects generation model: We assume that the objects observed in the training set are sampled from $\mathcal{X}$ according to some (unknown) probability distribution on p_1. That is, for any $\mathbf{x} \in \mathcal{X}$, $p_1(\mathbf{x})$ is the probability that $\mathbf{x}$ is the next object 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 $t = f(\mathbf{x})$

Let us restrict ourselves, in the following, to the second approach described above. Then, some concepts are relevant.

The quality of a predictor h, such as the one returned by the learner, is evaluated in terms of risk.

Given any element $\mathbf{x} \in \mathcal{X}$, the error of h when applied to $\mathbf{x}$ derives from the comparison of its prediction $h(\mathbf{x})$ and the correct target label y.

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

The error of a prediction $\hat{y} = h(\mathbf{x})$ is then defined in terms of prediction risk as given by applying the loss

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

In the more general case when only a probabilistic relation $p_2(y|\mathbf{x})$ (instead of a function f) is assumed between an item and the corresponding label, this corresponds to the expectation of the loss function with respect to such distribution

$$\mathcal{R}(\hat{y}, \mathbf{x}) \triangleq E_{p_2}[L(\hat{y}, y)] = \int_y L(\hat{y}, y) \cdot p_2(y|\mathbf{x}) dy$$

or, in the case of classification

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

1.3 Bayes estimator

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

$$y^*(\mathbf{x}) \triangleq \operatorname*{argmin}_{\hat{y}} \mathcal{R}(\hat{y}, \mathbf{x}) = \operatorname*{argmin}_{\hat{y}} L(\hat{y}, f(\mathbf{x}))$$

In the general case, this corresponds to

$$y^*(\mathbf{x}) \triangleq \operatorname*{argmin}_{\hat{y}} E_{p_2}[L(\hat{y}, y)] = \operatorname*{argmin}_{\hat{y}} \int_y L(\hat{y}, y) \cdot p_2(y|\mathbf{x}) dy$$

The predictor y^* is denoted as Bayes estimator.

However, observe that this approach cannot be applied since the function f (and the distribution $p_2(y|\mathbf{x})$) are assumed unknown.

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

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

In the general case,

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

Since p_1 and f (or p_2) are not known, the risk can only be estimated from the data available (the training set $\mathcal{T}$). This leads to the definition of empirical risk $\overline{\mathcal{R}}_{\mathcal{T}}(h)$, which provides an estimate the expectation of the loss function as the average loss on the training set.

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

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

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.

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

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

2 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 $p(t = 1|\mathbf{x}) = \frac{1}{2}$ 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:

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

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

Statistical learning theory studies from the theoretical point of view the effect of the chosen set of hypotheses and of the training set size on the quality (in terms of risk) of the predictor derived from learning.

3 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

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

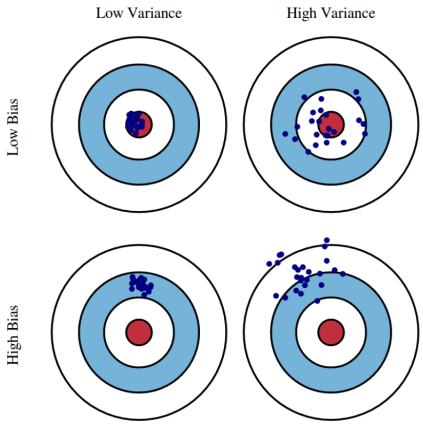


Figure 2: Graphical representation of bias and variance

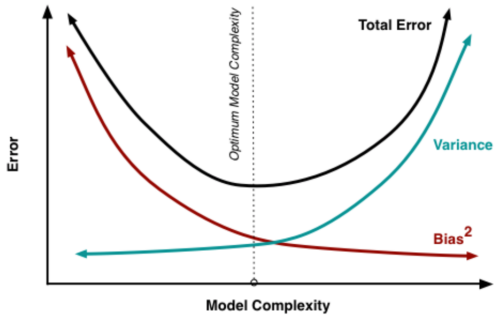


Figure 3: Bias and variance vs model complexity