Data Mining: Prediction Methods

Alfonso Urso, Antonino Fiannaca, Massimo La Rosa, Laura La Paglia, and Riccardo Rizzo, National Research Council of Italy, High Performance Computing and Networking Institute (ICAR-CNR), Via Ugo La Malfa, Palermo, Italy

 doi:10.1016/B978-0-323-95502-7.00060-9

Abstract

The classification methods presented in the "Data Mining: Classification and Prediction" chapter construct a model learning from a training data set and then uses it to classify new unseen instances. These methods are referred as *eager* learners. In this chapter will be introduced other classification methods, such as *k*-nearest-neighbor, case-based reasoning, genetic algorithms. Moreover, prediction methods will be explored, in particular referring to linear and nonlinear regression and finally two cases of generalized linear models: logistic and Poisson regression.

Key Points

- Understanding the theory and the working of the most common regressors algorithms.
- Presenting the main characteristics and drawbacks of each function estimator algorithm.
- Allowing the reader to choose the algorithm that best suits their needs.

Introduction

The chapter introduces different classification and prediction methods, whose learning models learn from the training data set and then use it to classify new unseen instances, highlighting for each model features, and advantages. The approaches discussed in the chapter are the following: k-nearest neighbor, genetic algorithm, and regression models.

The k-nearest neighbor algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier. It is based on the principle of finding a predefined number of training samples closest in distance to the new point and predict the label from these.

The genetic algorithm is another learning method used in classification and regression tasks, inspired by the process of natural selection of biology evolution. Likewise, the genetic algorithm belongs to the larger class of evolutionary algorithms (EA).

The chapter will also introduce to linear and non-linear regression methods, and some generalized linear models such as logistic and Poisson regression. The chapter aims to guide the reader in introducing different learning models used for classification and regression tasks, focusing also on practical examples.

k-Nearest Neighbor (*k*-NN)

A different approach to classification or regression is possible without any model construction first. This is known as the *lazy approach* because of the working procedure: the learner does not construct any general model before seeing the test instance. The learner stores all training instances and, only when a given test instance arrives, it generalizes the information coming from the examples to classify or predict the new class label or numeric value. *k–nearest neighbor* (*k-NN*) and *case-based reasoning* are examples of lazy learner.

k-NN method is based on the nearest neighbor decision rule (Fix and Hodges, 1951, 1952; Cover and Hart, 1967), according to which the class of an object never seen before by the classifier depends on a set of previously classified records. The nearest neighbor rule is widely applied in problems of pattern recognition, text categorization, ranking models, object recognition and event recognition (Bhatia, 2010).

How it Works

k-NN compares an unknown instance with those belonging to the training set and assigns it the class according to the similarity with training instances. Consider a training set of instances as pairs $(\boldsymbol{X}_i; y_i)$, where $\boldsymbol{X}_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ is a tuple[1] described by n attributes and y_i is the corresponding class label (qualitative or quantitative)[2]; $i = 1, \ldots, p$ is the total number of tuples in the data set. A tuple is represented by a point in a n-dimensional space of attributes (or feature space[3]). A new instance to be classified will

[1]A tuple is a sequence of values concerning respectively fixed attributes. It is a vector in the feature space.
[2]See the paragraphs 1 and 2 of the chapter "Data Mining: Classification and Prediction".
[3]The feature space concept was introduced in the chapter "Data Mining: Classification and Prediction", section "Components of a Generalized Linear Model".

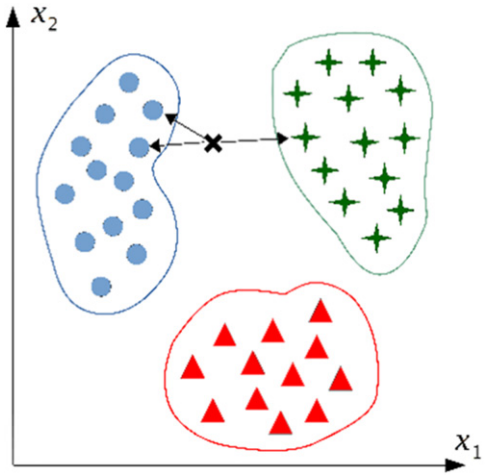

Fig. 1 *k*-NN classification in a 2-dimensional feature space; $k = 3$. The black cross represents a new instance to classify. The arrows mark the distance among the new tuple and 3 nearest neighbors.

take one among the class label of k training tuples that are nearest neighbors in the space of representation (see **Fig. 1**). By using a distance metric, the classifier establishes the neighbors in the feature space.

It is possible to use several *distances functions* to compute the similarities among tuples. This choice depends on the kind of the data set, if it includes numeric, symbolic or categorical attributes.

Distance functions

For numeric attributes, common distances are *Manhattan* and *Euclidean* distances, which represent two particular cases of the more general *Minkowski* distance.

Manhattan distance d_M between the tuples $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ is known also as Minkowski distance of order 1 and is given by the Eq. (1)

$$d_M(\boldsymbol{X}_1, \boldsymbol{X}_2) = \sum_{i=1}^{n} x_{1i} - x_{2i}. \tag{1}$$

The Euclidean distance d_E between two tuples $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ is computed as shown in Eq. (2):

$$d_E(\boldsymbol{X}_1, \boldsymbol{X}_2) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2}. \tag{2}$$

It is known also as Minkowski distance of order 2.

Hamming distance is used to compare two tuples expressed by strings of n characters (symbolic attributes). In this case, the distance reports the total number of correspondent positions where the symbols are different. For example, considering the strings "ATCG" and "ATGG", its Hamming distance is $d_M = 1$, differing only in the third position.

The computation of their distance for categorical attributes is not straightforward. The distance or, conversely, the similarity can be evaluated considering schemes of numeric "translation". Starting from the trivial case, the similarity between two categorical attributes is 1 if they are identical or 0 if they are different. Considering instances described by more than one categorical attribute, the similarity between them will increase with the number of matching attributes. This is the so-called *overlap* measure (Stanfill and Waltz, 1986). Additional sophisticated procedures allow evaluating differential grading of similarities between instances described by more than one categorical attribute (Boriah *et al.*, 2008). Other distances can be used, for example, *Jaccard* distance (Levandowsky and David, 1971), *Dice's* coefficient (Dice, 1945), Tanimoto coefficient (Tanimoto, 1958) and cosine distance (Qian *et al.*, 2004).

The k value

Once calculated the distances from the other training examples, *k*-NN classifier establishes the class of the new instance by choosing the most frequent label among k nearest neighbors (see **Fig. 2**). In prediction task, for a new instance in input, *k*-NN predictor returns a real value, which is the average value of k numeric labels of nearest neighbors.

The choice of the k value influences the performance of the *k*-NN classifier. Indeed, a too small value of k makes the classifier more sensitive to noising data. Conversely, if k value is too large, the classifier will consider also examples from other classes, possibly deviating the result of classification. In the limit case, all the training instances will be considered and the most frequent class label will be assigned to the new instance.

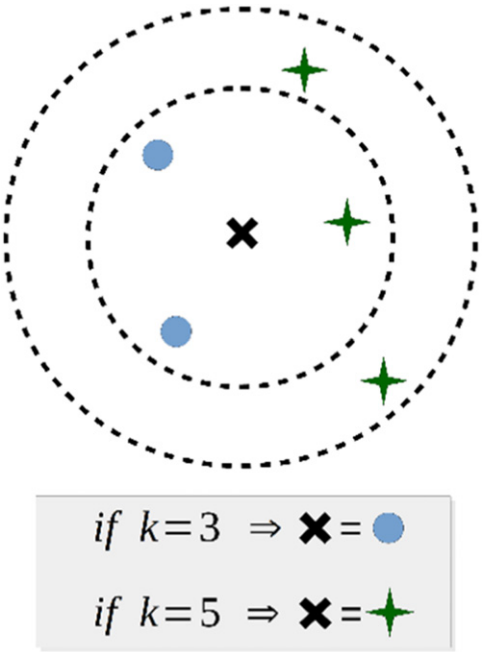

Fig. 2 The choice of the k value conditions the output of the classifier, including more or less example of different class.

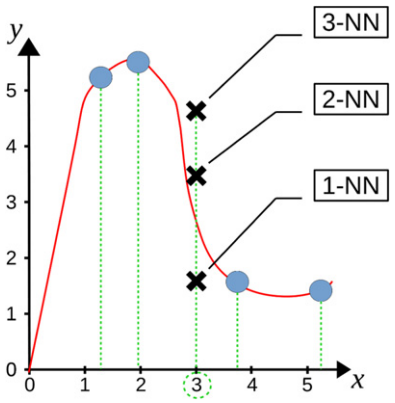

Fig. 3 k-NN regression: blue dots represent the training set plotted in a graph of y_j values against the feature values x_j. The red curve f is the true function underlying the training set. The regression method approximates the true (unknown) function. Therefore, given the new instance $x = 3$, it provides as output a numeric value $\hat{y}$ (one black cross for each value of k) that should be as near as possible to the true value $f(x)$. $\hat{y}$ value changes for different values of k.

The good value for the number k of neighbors is experimentally determined by estimating the error rate of the classifier/predictor with a start number $k^{(0)}$. If the performance needs to be improved, the number $k^{(0)}$ is increased, according to the number of new neighbors. The error rate can be estimated through probabilistic methods (Duda, Hart and Stork, 1991).

k-NN Regression

For dataset of instances $(\boldsymbol{X}_j; y_j)$ described by numeric attributes as component of the vector $\boldsymbol{X}_j$ and corresponding numeric value y_j, k-NN works as a regression method. Therefore, it allows to predict the numeric value $\hat{y}$ corresponding to an unseen instance vector $\boldsymbol{X}$, on the basis of the information of k nearest neighbors in the training set. Consider the simple case of dataset described by a single feature. Hence, the $j-th$ instance of the dataset in represented by the pair $(x_j; y_j)$, where x_j is a scalar. The entire dataset can be plotted in a 2-dimensional graph, where the x-axis represent the 1-dimensional feature space (see **Fig. 3**).

Given a new instance $(x = 4; \hat{y} = ?)$, k-NN provides as output a number $\hat{y}$ arising from the Eq. (3) (Hastie *et al.*, 2008):

$$\hat{y} = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i \tag{3}$$

where $N_k(x)$ is the set $\{x_1, \ldots, x_k\}$ of the training neighbors of the new value x. Clearly, $N_k(x)$ depends on the value chosen for k. Referring to the example depicted in **Fig. 3**, for $k = 1$, $N_k(x)$ will contain only one instance, which is the first in the list of the nearest neighbors. Consequently $\hat{y} = y_1 = 6$. For $k = 2$, $\hat{y} = \frac{1}{2}(6 + 3) = 4.5$ and so on. As shown in the **Fig. 3**, the suitable choice of k dictates the performance (with $k = 3$ the predicted value moves away from the true value).

Unlike the regression methods described below, k-NN regression is said to be a "non-parametric" method, since it does not assume any explicit form for the prediction function f, meaning that it does not construct a model for f.

Conclusive Remarks

As resumed in the work of Bhatia (2010), k-NN technique includes advantages like a very fast training and a simple learning. It shows robustness to noisy training data coming from pruning noisy tuples (Aha, 1992). On the other hand, the main disadvantages concern mainly the computational complexity (Preparata and Shamos, 1985) and memory limitation of this technique. The slowness of classification can be improved by using a partial distance calculation, where only a subset of attribute is involved in the compute of the distance (Gersho and Gray, 1992). The editing of the stored tuples removes useless training examples, helping to speed up the process of classification (Hart, 1968). Furthermore, the performance of k-NN algorithm can be biased by the choice of the k value (Guo *et al.*, 2003). Irrelevant attribute can easily mislead the decision of k–NN. For this reason, it is made more robust by a specific pruning on data tuples.

Genetic Algorithms Classifier

Genetic algorithms (GAs) (Holland, 1975) are stochastic algorithms used to solve optimization and search problems, (see Michalewicz, 1992). They are a type of evolutionary computation technique (Rechenberg, 1973) inspired by adaptation principles of natural selection. They are domain independent methods, therefore, GAs are applied in several fields of computer science, for example as optimization process in a number of problems such as routing and scheduling (the traveling salesman problem), game-playing, cognitive modeling, transportation problem and control problems (Janikow, 1993; Goldberg, 1989; DeJong, 1985) and also in the field of Neural Networks (Rojas, 1996).

Vocabulary

GAs works on a set of possible solutions to a given problem, by applying stochastic methods that simulate natural ways of evolution. The biological world influences both computing methods and the language of GAs. In biological language, the genetic information of each individual is contained in *chromosomes (see* **Fig. 4**, left side). A chromosome consists of units or *genes that control hereditariness of characters. Each gene occupies a given position or locus on the chromosome. The same gene can be in several states or alleles, entailing the different ways in which an individual's character (for example eyes color) manifests itself.*

In GAs, the chromosomes represent candidate solutions to a given problem. A chromosome is made up of a sequence of genes (see **Fig. 4**, right side). The genes encode a particular characteristic or feature of the candidate solution. For example, considering a

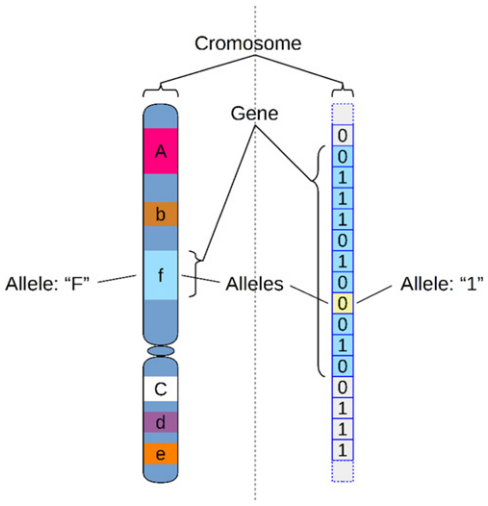

Fig. 4 left) Biological representation of a chromosome: a string of genes (bands with letters), which are a portion of DNA. Each portion occupies a given locus on the chromosome. Uppercase or lowercase letter in a locus represents the allele of the same gene; right) Computational representation of a chromosome: a sequence of genes, which are codified by one or more bits. The possibilities "0" or "1" at the same locus give rise to alleles of the same gene.

problem of optimization, the optimization function is a chromosome. Each parameter of the optimization function is a gene, encoded by either a single bit or a short block of bits. A gene represented by single bit can be in two states, either "0" or "1". These two possibilities are the alleles of that gene. The different alleles of the same gene represent different values of the same feature in GA vocabulary. For a gene that encodes more than two possible alleles, a block of bits is used.

How it Works

The basic idea of GA is to find the best individual in a population of chromosomes, namely candidates' solutions of a given task (optimization function or classification rule), following a process of natural selection. During such process, the fitter individuals of the population are chosen to create the *offspring. The offspring will result in a new successive population replacing the previous one. Each replacement gives rise to a new generation, potentially including the fittest one among several individuals and hence the best solution to the problem.*

GAs employ the way of natural selection to allow proliferation and modification of good solutions and inhibit bad solutions, in perfect analogy with the biological environment (Chang and Lippmann, 1991).

The implementation of GA is based on its *genetic operators. The simplest form of GA employs creation, selection, crossover and mutation. Genetic operators will be discussed in the following section. Furthermore, an evaluate function* (Michalewicz, 1992) or *fitness function* (Mitchell, 1996) allows evaluating each individual assigning it a score, namely the *fitness. The fitness of an individual depends on its goodness as a solution for the considered problem. The candidate who approaches the best solution will have the highest score. The following section introduces an example of the fitness function and a simple scheme of GA with genetic operators.*

Fitness function

In the optimization problem, the aim is to find the set of parameters that maximize or minimize a given function. Consider for example the function of Eq. (4) (Riolo, 1992):

$$f(x) = x + \sin(32x), with 0 \leq x \leq \pi \tag{4}$$

The values of x maximizing the function are the candidate solutions. The possible values of x are encoded in strings of bit: they represent the chromosomes on which will operate the GA. To evaluate a candidate bit string, it is enough to convert it in the correspondent real number x' and calculate the value of the function in x'. This value is the fitness of the string.

Operators of GA: Creation, selection, crossover, mutation

The evolution process in GAs starts with the *creation of the initial population by the specific homonym genetic operator. The individuals in a population are chromosomes encoded by the bit string that is initialized to "1" or "0" values.*

Once the first population is created, the fitness of individuals is evaluated. Then, the *selection operator identifies the fittest candidates to breed. The probability of selection is indeed an increasing function of fitness* (Mitchell, 1996). The same chromosome can be selected more times to reproduce if it is fitter than others.

A *crossover operator acts on a couple of selected chromosomes, the parents, exchanging portions of these, In* **Fig. 5**, it is shown the simplest crossover operator, which acts in a single locus of the chromosome (single point). First, the crossover operator randomly chooses a locus on the parents' chromosomes. Then it exchanges the substrings, creating two offspring. The aim of this process is to mix the parents to produce new chromosomes.

Crossover can provide new chromosomes until that the individuals are not too similar to each other. For this scope, the new individual can be altered by the operator *mutation that randomly selects bits in a string and then inverts them, as shown in* **Fig. 5**.

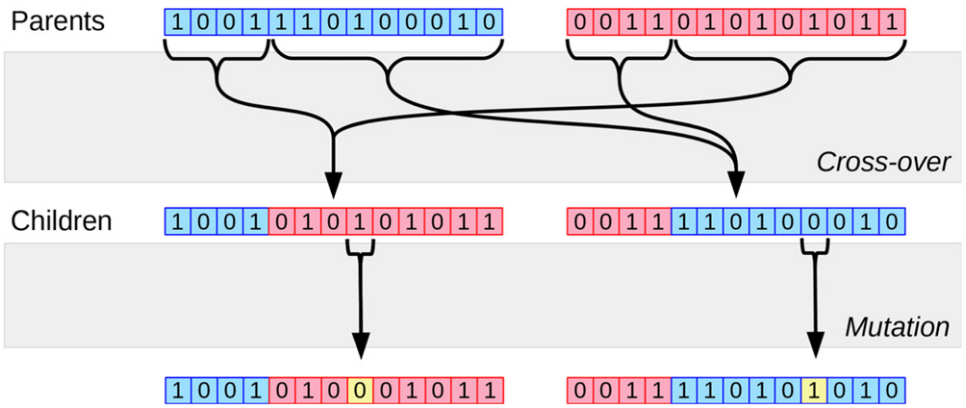

Fig. 5 The parents "blue" and "pink" strings breed through the crossover operator. It creates two new chromosomes children (offspring) from the single crossover point at the fourth-bit position. The mutation operator changes the offspring.

Table 1 Pseudocode summarizing the steps of a generic GA

```
1  Creation of a random population of n chromosomes
   (initialization of bit strings);
2  Evaluation of the fitness of each chromosome;
3  Intermediate step (to repeat until creating n offspring):
      a) Selection of two chromosomes as parents (based on step 2)
      b) Crossover with probability p_c
      c) Mutation with probability p_m
4  Replacement of the current population with new population;
5  Go to step 2.
```

Crossover and mutation are random operators, meaning that they will act with a fixed probability, respectively p_c (crossover probability or crossover rate) and p_m (mutation probability or mutation rate).

A simple scheme of operation of GA is illustrated below.

Scheme of a simple GA

The process begins with the creation operator, which produces an initial population of chromosomes. The intermediate step provides a recursive modification of population, by applying other genetic operators such as selection, crossover and mutation. At the end of the evolution process, only the fittest offspring will survive, representing the selection of the best solutions.

The following scheme shown in **Table 1** sums up the steps of a GA observing what has been reported by Mitchell (1996):

The iterations of this procedure are called *generations. Their number varies typically from 50 to 500 or more (*Mitchell, 1996). The set of generations is called *run. After a run, the fitness function will indicate one or more individuals highly fit in the population. The performance of the GA often depends on the characteristic chosen to treat a given problem, for example, the size n* of population and p_c and p_m value. Considering the importance of randomness in GA, a better information can result from the average over many different runs of the given problem.

Optimization Problems and Classification Tasks

In optimization problems, the individuals in a population are the possible solutions (Sivanandam and Deepa, 2007). Fitness function assesses each solution by a fitness value (Chang and Lipmann, 1991).

Genetic search algorithms show a suitable approach to feature creation, feature and examples selection. These skills make GAs suitable for pattern classification tasks (Chang and Lippmann, 1991).

In classification problems, the individuals in the initial population are rules randomly generated. Classification accuracy evaluates the fitness of a rule, considering a set of training instances (Han *et al.*, 2012). A GA can be used to optimize the parameter set of a prediction function by using the error rate of the prediction function as fitness value (Johnson *et al.*, 2014).

Conclusive Remarks

The main advantages of GAs are robustness and independence of their search mechanism from the application field. In particular, domain independence allows that a new employee needs only a proper encoding of the given problem (Janikow, 1993). In speech recognition domain, GAs have a processing time for features selection similar than traditional methods but use a lower number of input feature (Chang and Lippmann, 1991). GAs can work on tasks of artificial machine-vision producing outputs of classification with error rate nearly 0%, obtaining results better than KNN and Neural Network Classifiers (Chang and Lippmann, 1991). The ease of application is also an advantage of GAs, in addition to the effectiveness in finding a proper solution to in-depth search problem (Chang and Lippmann (1991)).

On the other hand, the run time of GAs can be long (Chang and Lippmann, 1991; Janikow, 1993), this being one of major drawbacks of this technique. Furthermore, domain independence entails as a disadvantage that the performance of GAs is heavily dependent on the quality of the problem coding.

Linear and Nonlinear Regression Prediction

The aim of this paragraph and the successive one is to introduce numerical prediction methods,[4] which answers to given task by supplying continuous target values. Problems that require numerical predictors can be addressed by a *function estimator* or *regressor* (Flach, 2012). It provides real numeric values starting from a set of numeric input data such as observations or measurements. This kind of predictor builds a *model function* that fits training data of a given task. Such a function will be able to predict unknown numerical data for a new instance pertaining to the same domain of the learned problem (Han *et al.*, 2012).

Regression predictors forecast data trends by applying the statistical tool of regression analysis in several domains of interest from marketing management (Stock and Watson, 2003) to bioinformatics analysis (Wu *et al.*, 2009).

Regression is an approach mostly used for numerical prediction. According to the kind of the problem to treat, two different models can be used, linear and nonlinear regression. Both models describe a relationship between a dependent variable y and one or more independent variables $\boldsymbol{X}$. The main difference between these two types of regression relies on the relationship between the independent variable and the parameters of regression, i.e. if it is a linear combination of those or not.

The curve obtained from regression analysis is straight-line for linear regression and can be of other types for nonlinear regression (polynomial curves with a degree greater than 1 are considered a special case of multiple regression and can be treated with linear regression; they are an example of functions called linear regression models) (Flach, 2012; Bishop, 2006).

How Regression Prediction Methods Work

Regression prediction methods fall into the category of *geometric models*. Here, the techniques represent a given task and therefore data set in an n-dimensional geometric space, called feature space,[5] being each instance described by n features. Regression methods construct classification models by employing geometric concepts such as lines or planes or considering measures of distance as similarities among properties.

To train a regression prediction method, it is necessary to model training data with a function f. Training data are a set of examples, where each instance is described by a pair $(\boldsymbol{X}, y)$, $\boldsymbol{X} = (x_1, \ldots, x_n)$ is the vector of the features and y is the predefined numeric label representing a "true" function value. Starting from the training data set, a function f that maps an input vector of attributes $\boldsymbol{X}$ from the feature space to the space of real number will be built, providing $f(\boldsymbol{X})$ as output. The input $\boldsymbol{X}$ is the vector of the independent or predictor variables, whereas $f(\boldsymbol{X})$ is the dependent or *response variable* (Flach, 2012).

During the training process, the predictor learns the best function that fits the data, identifying the suitable parameters $\boldsymbol{w} = (w_1, \ldots, w_n)$for f. The learning relies on the difference or *residual* (ϵ_i) between the actual and the predicted values, for the $i-th$ instance respectively y_i and $f_{\boldsymbol{w}}(\boldsymbol{X}_i)$, as shown in the Eq. (5)

$$\epsilon_i = y_i - f_{\boldsymbol{w}}(\boldsymbol{X}_i) \tag{5}$$

In particular, to avoid that positive and negative errors mutually cancel out, it is usual to consider squares of residuals $(\epsilon_i)^2$ to evaluate regression models. Therefore, the *error function*[6] is expressed from the *residual sum-of-squares* $RSS(\boldsymbol{w})$, given by $\sum_{i=1}^{p} \epsilon_i{}^2$, as shown in the following Eq. (6):

$$RSS\left(\boldsymbol{w}\right) = \sum_{i=1}^{p} \left(y_i - f_{\boldsymbol{w}}(\boldsymbol{X}_i)\right)^2 \tag{6}$$

where the sum run over the total number p of instances in training set.

The *method of the least squares* is used to find the best function, minimizing the error function RSS with respect to the parameters $\boldsymbol{w}$. The function $f_{\boldsymbol{w}}$ looks for this procedure will be bound to satisfy the relation of minimum squares of residuals, and it will be hence as close to the data as possible.

Linear Regression Predictor

Linear is the simplest type of regression. The model is a linear function of a predictor variable $\boldsymbol{X}$. When a scalar predictor variable is used rather than a vector, the independent variable $\boldsymbol{X}$ is a single number x. In this basic case, known as *simple linear regression*, it is hypothesized that the model function describing each instance i of the data is straight-line, expressed by the Eq. (7).

$$f_{\boldsymbol{w}}(x_i) = w_0 + w_1 x_i \tag{7}$$

where the weights w_0 and w_1 are respectively the intercept and the slope of the line, also known as *regression coefficients* or *regression parameters*. To get the coefficients that are suitable for the given problem, $RSS(\boldsymbol{w})$ will be minimized. Indeed, according to the least squares method, the weights values will be obtained by solving the system of two *normal* Eqs. (8) and (9) which put equal to zero the partial derivatives of RSS with respect to the parameters weights w_0 and w_1:

[4]See also the first two paragraphs of the "Data Mining: Classification and Prediction" chapter.
[5]Introduced in section "Linear and nonlinear classifier" of the "Data Mining: Classification and Prediction" chapter.
[6]The error function was introduced in the chapter "Data Mining: Classification and Prediction".

$$\frac{\partial RSS}{\partial w_0} = -2\sum_i^p (y_i - w_0 - w_1 x_i) = 0 \tag{8}$$

$$\frac{\partial RSS}{\partial w_1} = -2\sum_i^p (y_i - w_0 - w_1 x_i) x_i = 0 \tag{9}$$

The system leads to the solutions shown in Eqs. (10) and (11):

$$w_1 = \frac{\sum_{i=1}^{p} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{p} (x_i - \bar{x})^2} \tag{10}$$

and

$$w_0 = \bar{y} - w_1 \bar{x} \tag{11}$$

where $(x_i; y_i)$ is one of the p pairs of data set, and $\bar{x}$ and $\bar{y}$ are the mean value of the set $\{x_i\}$ and $\{y_i\}$ where $i = 1, 2, \ldots p$.

In training phase, the predictor finds the weights w_1 and w_0 and therefore the builds the function f that models the data set. Linear regression predictor will employ this function to make a numerical prediction about a new numeric instance x' as input in the Eq. (11).

Multiple linear regression

When the independent variable is a vector $\boldsymbol{X} = (x_1, \ldots, x_n)$, the Eq. (11) changes in Eq. (12):

$$f_{\boldsymbol{w}}(\boldsymbol{X}) = w_0 + \sum_{j=1}^{n} w_j x_j \tag{12}$$

where the vector of weights $\boldsymbol{w}$ expresses the regression coefficients (or parameters). Now, the curve that fits the data change from a straight-line to a plane (for $n = 2$) or a more general hyperplane (for $n > 2$). Each component of the vector $\boldsymbol{X}$ is a predictor variable, and hence the regression is defined *multiple* instead of *simple*, see previously, where a single scalar predictor variable x is used. Even in this case, the residual sum-of-squares $RSS(\boldsymbol{w})$, expressed by the Eq. (6) have to be minimized with respect to the parameters $\boldsymbol{w}$ to obtain their best values, according to the minimal squares method.

Polynomial regression

Polynomial regression adopts the function $f_{\boldsymbol{w}}(x)$, defined in the Eq. (13):

$$f_{\boldsymbol{w}}(x) = w_0 + w_1 x + w_2 x^2 + \ldots + w_m x^m = \sum_{j=1}^{m} w_j x^j \tag{13}$$

where m is the order of the polynomial and x is the only predictor variable. Although the polynomial curve is a nonlinear function of the independent variable x, it is a linear combination of the polynomial coefficients $\boldsymbol{w}$. Such kind of functions can be treated as linear regression. Indeed, linear regression is used for models that are linear in the parameters (are in the predictor variables linear or not).

The least squares method can be applied to estimate the parameters, resorting to the technique of multiple regression. Indeed, it is possible to consider a set of new variables as expressed in Eq. (14):

$$x^j = x_j \tag{14}$$

where $j = 1, \ldots, m$. Replacing these new variables in Eq. (13), each term of order higher than one is substituted with a new independent variable and the polynomial function takes the form of Eq. (14), resulting in a linear form of the multiple predictor variables.

Lasso regression

The "Least Absolute shrinkage and Selection Operator" (Lasso) regression technique is an extension of the basic linear regression (Tibshirani, 1996). Lasso integrates either feature selection and regularization. Unlike linear regression, Lasso aims at minimizing the following objective function:

$$\sum_{i=1}^{p} \left(y_i - w_0 - \sum_{j=1}^{n} w_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{n} w_j = RSS + \lambda \sum_{j=1}^{n} w_j \tag{15}$$

where p is the number of training instances and n is dimension of the feature vector. The first term is the residual-sum-of squares (see Eq. (6)), while the second one is the L1 norm penalty term, since the L1 norm of a coefficient vector w is defined as:

$$w_1 = \sum_{j=1}^{n} w_j \tag{16}$$

The penalty term allows some of the regression coefficients to go exactly to zero, pushing this way the sparsity of the model. Finally, λ is a tuning parameter that represents the strength of the penalty: greater λ values means more coefficients becoming zero. The penalty term, then, let the lasso regression to perform feature selection, because it drives some coefficients to become exactly zero when the tuning parameter is large enough. The features associated to zero coefficients, in fact, are considered not so relevant to the target variable, with the penalty terms able to identify and rank the most important features. This mecanism is managed by the tuning parameter λ. The tuning parameter, in fact, controls the impact of the penalty term and determines the tendency of the regression coefficients β. That means λ is related to the regularization of the model, with larger λ values leading to strong regularization, and small λ values pushing towards weak regularization.

It is noteworthy that the choice of the tuning parameter λ is a critical choice when adopting a lasso regression model. Its choice usually depends on the current available dataset and on the desired sparseness degree of the model. Some particular training procedures, such as cross validation, can allow to try different λ values and search for the one providing the best model scores.

Ridge regression

Ridge regression (Hoerl and Kennard, 2000) is another kind of linear regression method that considers some learning issues such as multicollinearity and overfitting. Similar to lasso regression, ridge regression aims at minimizing the following objective function:

$$\sum_{i=1}^{p}\left(y_i - w_0 - \sum_{j=1}^{n} w_j x_{ij}\right)^2 + \lambda \sum_{j=1}^{n} w_j^2 = RSS + \lambda \sum_{j=1}^{n} w_j^2 \tag{17}$$

where p is the number of training instances and n is dimension of the feature vector. As seen in Eq. (17), the first term is the RSS, while the second term is the L2 norm penalty term, since L2 norm is defined as:

$$w_2 = \sqrt{w_1^2 + w_2^2 + \ldots + w_n^2} \tag{18}$$

λ is a tuning parameter that represents the strength of the penalty. Differently from lasso regression, in this case large value of λ tends to reduce the values of the coefficients towards zero, but not exactly equal to zero. Once again, therefore, the tuning parameter λ is responsible for regularization of the model. This behavior is able to prevent the so-called multicollinearity, that occurs when the independent variables of a regression model are actually correlated. On the other hand, since ridge regression does not put to zero the coefficients, but just mimise them, it is not well suited for feature selection. Finally, ridge regression provides a balanced model, because the L2 penalty term tends to estimate small weights for variable associated to great weights, rather than to assign zero values to variables related to small weights.

Elastic net

Elastic Net is a regression technique that combines the penalty terms introduced by both lasso and ridge regressions (Zou and Hastie (2005)). Elastic Net, in fact, aims at minimizing the following objective function:

$$\sum_{i=1}^{p}\left(y_i - w_0 - \sum_{j=1}^{n} w_j x_{ij}\right)^2 + \lambda_1 \sum_{j=1}^{n} w_j + \lambda_2 \sum_{j=1}^{n} w_j^2 =$$

$$= RSS + \lambda_1 w_1 + \lambda_2 w_2^2. \tag{19}$$

In this case, we have two tuning, or regularization, parameters λ_1 and λ_2 that control the L1 penalty and the L2 penalty, respectively. From Eq.(19) we can see how elastic net a general case of lasso and ridge regression methods is. If $\lambda_1 = 0$ we have ridge regression, whereas if $\lambda_2 = 0$ we have lasso regression. Eq.(19) can be rewritten in order to consider just one tuning parameter λ as follows:

$$RSS + \lambda * (1 - L1Ratio) \sum_{j=1}^{n} w_j^2 + \lambda * L1Ratio \sum_{j=1}^{n} w_j \tag{20}$$

where L1Ratio represents the amount, in percentage, of the L1 penalty, scaled according to the value of λ. For example, regarding a given λ value, if we want a 30% of the L1 penalty, the corresponding L2 penalty will be equal to λ *(1-L1Ratio) = λ*0.7. As seen earlier, with L1Ratio = 0 we have the ridge regression and with L1Ratio = 1 we have the lasso regression.

Elastic net, integrating both lasso and ridge penalty terms, introduces some advantages. First of all, it can perform feature selection because the coefficients of less significant or irrelevant variables can be set to zero. A model with fewer variables, moreover, is less exposed to possible overfitting. Elastic net is a more robust model with regards to ridge and lasso because it embed both of them and it is also able to cope with multicollinearity, thai is correlated variables, and variables with different scales. Finally, elastic net provided better performances than other linear regression models, above all when dealing with dataset with a large number of features.

PLS regression

Partial Least Squares Regression (PLS Regression) is a powerful multivariate statistical technique (Tenenhaus, 1998) that finds its application in several scientific fields, such as chemometrics, bioinformatics, and process modeling, because it can extract relevant information from complex and collinear datasets.

PLS Regression operates on the principle of extracting latent factors that capture the maximum covariance between the predictors and the response variable. Unlike standard regression methods, PLS Regression does not rely solely on the original predictor variables but constructs a set of latent variables (factors or components) representing linear combinations of the original variables.

The main advantages of the PLS Regression algorithm (Russolillo, 2012) are listed in the following:

Dealing with Collinearity: PLS Regression gives good results in scenarios where traditional regression models struggle due to multicollinearity. By constructing latent variables that summarize the information in the original predictors, PLS effectively addresses collinearity issues.

Effective in High-Dimensional Data: PLS Regression offers a robust solution in fields where the number of predictors far exceeds the number of observations. It efficiently handles high-dimensional datasets, making it well-suited for big-data scientific studies.

Simultaneous Analysis of Multiple Responses: PLS Regression extends naturally to multiple response variables, allowing scientists to simultaneously model complex relationships involving several dependent variables.

PLS regression standard algorithm

PLS Regression works on extracting latent factors that capture the maximum covariance between the predictors and the response variable. Unlike standard regression methods, PLS Regression does not rely solely on the original predictor variables. Still, it constructs a set of latent variables (factors or components) representing linear combinations of the original variables. This helps address multicollinearity and high dimensionality issues, which are prevalent in many scientific datasets.

PLS Regression is composed of four steps:

(1) **Data Standardization:** Before applying PLS Regression, it is common practice to standardize the data to ensure that all variables have a mean of zero and a standard deviation of one. This step is crucial for preventing larger-scale variables from dominating the modeling process.
(2) **Construction of Latent Variables:** PLS Regression decomposes both the predictor matrix $\boldsymbol{X}$ and the response matrix $\boldsymbol{Y}$ into sets of latent variables. These latent variables are created iteratively, with each new component capturing the maximum covariance between X and Y while being orthogonal to the previously constructed components.
(3) **Weighting Scheme:** PLS Regression introduces a weighting scheme during decomposition, assigning weights to the predictor variables based on their contribution to the covariance structure. This weighting helps in emphasizing the most relevant information in the data.
(4) **Model Calibration:** The final step involves using the latent variables to build a predictive model. The relationship between the original predictors and the response variable is captured through a set of coefficients, and the model can be used for prediction and interpretation.

The PLS regression model assumes the set of $\boldsymbol{P}$ predictor variables and the set of $\boldsymbol{R}$ response variables underly in a common structure that can be resumed by a few latent components t_h (h = 1 . . . H), calculated as a linear combination of the predictor variables.

Table 2 Pseudocode summarizing the steps of PLS Regression algorithm

```
Input: E_0 = X, F_0 = Y
Output: W, C, T, U, P
   for all h = 1, ..., H do
      Step 0: Initialize u_h
      Step 1:
         repeat
            Step 1.1: w_h = E'_{h-1}u_h/||E'_{h-1}u_h||
            Step 1.2: t_h = E_{h-1}w_h/(w'_h w_h)
            Step 1.3: c_h = F'_{h-1}t_h/(t'_h t_h)
            Step 1.4: u_h = F_{h-1}c_h/(c'_h c_h)
         until convergence of w_h
      Step 2: p_h = E'_{h-1} t_h/(t'_h t_h)
      Step 3: E_h = E_{h-1} - t_h p'_h
      Step 4: F_h = F_{h-1} - t_h c'_h
   end for
```

Predictor and response are decomposed into:

$$X = \mathbf{T_H P'_H} + \mathbf{E_H}$$
$$Y = \mathbf{T_H C'_H} + \mathbf{F_H}$$

where matrices X and Y represent predictors and response matrices, P_H and C_H are the loading matrices, and E_H and F_H are the residual matrices representing the part of the variability in data due to noise.

The PLS regression algorithm, as presented in Tenenhaus (1998) is shown in **Table 2**:

PLS main implementations

Partial Least Squares algorithms come in two main implementations (Halstensen, 2020): PLS1 (Partial Least Squares Type 1) and PLS2 (Partial Least Squares Type 2). These variants differ in handling the relationship between the predictor variable X and the response variable Y in regression tasks.

PLS1 is designed for cases with a single response variable of interest. PLS1 constructs latent variables (components) that maximize the covariance between the predictor matrix (X) and the response variable (Y). Each latent variable is designed to capture the information shared between X and Y while orthogonal to previously constructed components. The primary focus is on the relationship between X and a single Y. PLS1 is commonly used when there is a clear and specific dependent variable of interest, such as in univariate regression tasks.

PLS2 extends the PLS algorithm to cases where there are multiple response variables. PLS2 constructs latent variables to maximize the covariance not just between X and a single Y (as in PLS1) but simultaneously maximizes the covariance between X and multiple Y variables. The latent variables capture shared information between X and a set of response variables. PLS2 is applied in scenarios where the relationships between multiple predictors and multiple responses need to be modeled simultaneously.

Anyway, PLS1 and PLS2 share some general features; both build latent variables iteratively, refining the model in each step to maximize the covariance between the predictors and the response(s). Also, in both variants, the constructed latent variables are orthogonal to each other, meaning that they capture unique information not explained by the previous components. Finally, similar to the standard PLS algorithm, PLS1 and PLS2 often begin with data standardization to ensure that all variables have a comparable scale.

In general, PLS1 and PLS2 find applications in various domains, including chemometrics, bioinformatics, and other fields where the relationship between multiple predictor variables and one or more response variables needs to be modeled.

PLS variants and extensions

Several variants and extensions of the Partial Least Squares (PLS) algorithm have been developed to address specific challenges or face different data mining scenarios, offering flexibility in addressing specific challenges and extracting valuable insights from complex datasets (Tenenhaus *et al.*, 2005; Trygg and Wold 2002; Embrechts and Ekins 2007; Chun and Keles, 2010). Some noteworthy PLS variants used in data mining are reported in the following:

(1) Sparse PLS Regression (sPLS) addresses the issue of model interpretability and reduces the number of variables in the model. It is used when feature selection is crucial, introducing sparsity constraints during the model fitting. This helps identify and focus on a subset of the most relevant variables, improving model interpretability and potentially enhancing prediction performance.
(2) Kernel-based PLS (K-PLS) extends PLS to handle non-linear relationships in the data by using kernel functions. K-PLS is applied in scenarios where linear models do not have well-captured relationships between variables. It transforms the data into a higher-dimensional space using kernel functions, making it possible to capture non-linear patterns.
(3) PLS-Canonical Analysis (PLS-CA) integrates PLS with canonical correlation analysis to jointly model relationships between two sets of variables. PLS-CA is used in data mining applications where understanding the relationships between two sets of variables is crucial. It helps identify latent structures that explain the maximum covariance between the two sets of variables.
(4) Orthogonal PLS (O-PLS) handles the problem of model interpretation and focuses on separating predictive and orthogonal components. O-PLS is employed when there is a desire to isolate predictive information from orthogonal variation in the data. This variant is helpful in situations where the goal is to enhance the interpretability of the model by distinguishing between systematic and noise-related variations.
(5) PLS-Path Modeling extends PLS for structural equation modeling to assess relationships between latent variables. PLS-Path Modeling is applied when the goal is to model complex relationships among latent variables. It combines the strengths of PLS with the structural equation modeling framework, allowing for a comprehensive exploration of relationships in a data-driven manner.

Nonlinear Regression

Nonlinear regression is employed to describe observational data by using a function $f_w(X)$ that is a nonlinear combination of the model parameters. In such function, the independent variables can be greater or equal to one. An example of nonlinear regression

curve is in the following Eq. (21):

$$f_{\boldsymbol{w}}(\boldsymbol{X}) = w_0 x_1^{w_1} x_2^{w_2} \tag{21}$$

where $\boldsymbol{w} = (w_0, w_1, w_2)$is the vector of the parameters and $\boldsymbol{X} = (x_1, \quad x_2)$ is the vector of independent variables.

For nonlinear regression, there is not a general method to obtain the best parameters. Indeed, the least squares method for nonlinear case leads to a system of normal equations without an analytic solution for $\boldsymbol{w}$. Therefore, numerical methods will be considered to solving nonlinear least squares.

In general, nonlinear regression entails an iterative procedure to estimate the best fit. A first fitting curve is generated from an initial value for each parameter. The *RSS* function, see Eq. (6), estimates the vertical distance between the data points and this curve. The next step tries to reduce such distance adjusting regression parameters iteratively, leading to a minimum of *RSS*. The obtained parameters depend on the choice of the initial values and also on the numerical optimization methods adopted to minimize *RSS*.

Examples of numerical methods used to minimize *RSS* are:

(1) the gradient descent (or steepest descent) method;
(2) the Gauss-Newton method;
(3) the Levenberg-Marquardt method;
(4) the Gaussian process regression.

Gradient descent (or steepest descent) method

This technique[7] exploits the information of the gradient of a function to find, step by step, the direction of steepest descent to its minimum value. The method begins with an initial arbitrary solution of the minimization problem and then changes it with a variation in the direction indicated by the gradient.

In the regression case, *RSS* is the function to minimize by varying the parameters $\boldsymbol{w}$. The method starts generating a first fitting curve resulting from an initial value $\boldsymbol{w}^{(0)}$ for the parameters. Then, it calculates the *RSS* between the actual value y_i of data set and the value resulting from the first fitting curve $f_{\boldsymbol{w}^{(0)}}(x_i)$. To minimize this *RSS* value, the initial solution $\boldsymbol{w}^{(0)}$ will be updated iteratively considering the relation $\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} + \alpha_k \boldsymbol{p}_k$, where k is the step of procedure iteration, α_k is the small amount of variation for the parameter $\boldsymbol{w}$, and $\boldsymbol{p}_k = -\nabla RSS(\boldsymbol{w}^{(k)})$ is the direction of steepest descent. *RSS* is computed at every new step, to monitor the procedure. The minimum will be reached when every further variation of $\boldsymbol{w}$ will produce an increase in the corresponding *RSS* value.

Gauss-Newton method

As for the previous method, Gauss-Newton starts from an initial estimate for the parameters $\boldsymbol{w}^{(0)}$. In the next step, it approximates the fitting function f as a function of the parameter w with a Taylor-series

about the point $\boldsymbol{w}^{(0)}$. At the 1st order, the fitting function is approximated to a line, according to the local linear model. Hence, *RSS*′ is computed considering the linear approximation of f and minimized following the usual linear squares methods. The resulting value for the parameters $\boldsymbol{w}^{(1)}$, is used in the next iteration of the method, until the entire procedure converges, i.e. when the original *RSS* is minimized (Ruckstuhl, 2010).

Levenberg-Marquardt

It consists of a blending of the previous methods, taking respective advantages. Indeed, the gradient descent works well in the initial iterations, but its performance decreases when parameters are near to the best values. On the other hand, Gauss-Newton method works better in successive iterations rather than in early ones. Levenberg-Marquardt starts its process by using a gradient descent approach and gradually changes it with the Gauss-Newton approach (Motulsky and Christopoulos, 2004).

Gaussion process regression

A Gaussian Processes Regression (GPR) model can make predictions by incorporating prior knowledge, represented by kernels, and provides uncertainty measures for these predictions (Rasmussen and Williams, 2006). In the context of GPR, when given a set of observed data points, the regression is conducted by Gaussian processes, defining a distribution across an infinite number of functions that can fit the given data points (Ghahramani, 2011). Specifically, GPR is a non-parametric Bayesian approach to inference. Instead of deducing a distribution over the parameters of a parametric function, Gaussian processes enable the inference of a distribution directly over the function of interest. The Gaussian process establishes a prior function, which transforms into a posterior function once specific values from the prior distribution are observed (Krasser, 2018).

Further considerations

Unlike the linear case, nonlinear regression can find a local minimum of the *RSS* curve that is not the "true" best values for the regression parameters. An example of a false minimum is shown in **Fig. 6**. Finding a local minimum does not depend on the numerical method chosen, since it can be determined by the initial choice of the parameters $\boldsymbol{w}^{(0)}$. To overcome this problem, it is advisable to

[7]It was introduced in the "Data Mining: Classification and Prediction" chapter, about the backpropagation algorithm of feedforward neural networks.

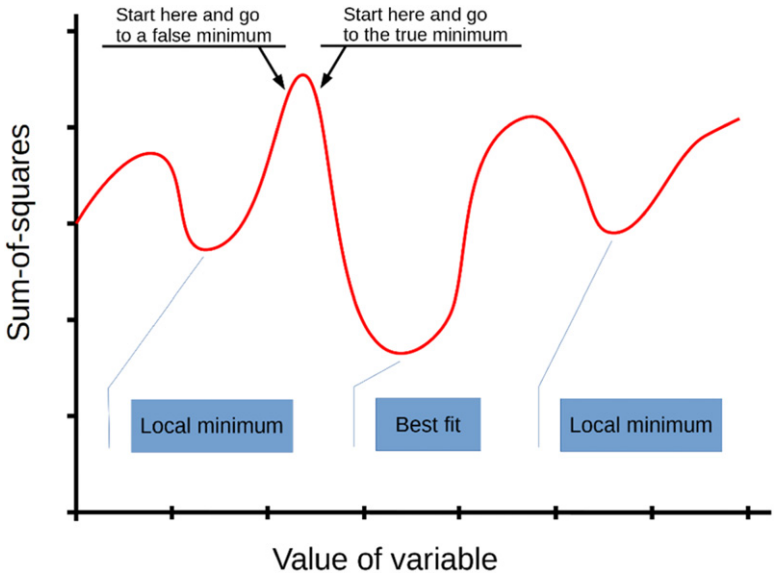

Fig. 6 Finding a local minimum is an intrinsic problem of nonlinear regression. It can be overcome by running nonlinear regression using different initial values for several computings.

repeat nonlinear regression many times with different parameters, i.e. using different initial values. The optimization process can guarantee the best fit solution if different choice of the vector $\boldsymbol{w}^{(0)}$ for the regression curve lead to the same minimum in *RSS* curve.

Goodness of a Model

Once the model is built, it can be useful to confirm the goodness of the fit. The meaning of the data, when it is known, suggests the type of regression to use and, as a general rule, the goodness of a fit relies on the acceptability of the parameters's value found. In general, to confirm the goodness of a fit, it is used the r^2or R^2 index (traditionally distinct for linear and nonlinear regression respectively) and the residual analysis.

r^2 or R^2

r^2 index is a number without units between 0.0 and 1.0. It is calculated through the Eq. (22):

$$r^2 = 1.0 - \frac{RSS_{reg}}{RSS_{tot}} \tag{22}$$

where RSS_{reg} is the same of the Eq. (6) and RSS_{tot} indicate the residual sum-of-squares of actual data point y_i from a horizontal line of equation $y = \bar{y}$, i.e. passing through the mean of all y_i values. r^2, therefore, states if the found function of regression fits the data better than a horizontal line, in which case the r^2 value is close to 1. The comparison is shown in **Fig. 7**:

R^2 have the same meaning of r^2 but is capitalized to indicate nonlinear case.

Residual analysis

The residual analysis allows assessing if the model chosen to describe the data is suitable or not. This analysis evaluates the trend of residuals ϵ_i of the Eq. (5) as a function of the independent variables $\boldsymbol{X}$. The **Fig. 8** illustrates three plots of residuals ϵ_i against the single independent variable x. A random pattern in the plot (left panel) confirms that a linear model is suitable for the regression analysis. Otherwise, another model would be used.

Conclusive Remarks

Method of regression rely on the assumption that the scatter of the data around the fitting line or curve follows a Gaussian or normal distribution. Therefore, the error of the dependent variable is random with mean zero. Furthermore, another hypothesis is that the variance of the error is a constant (*homoscedasticity*), i.e. the measure of the square variability of the error with respect to the mean of its distribution is uniform along the curve (Motuslsky and Christopoulos (2004)). Therefore, there should not be any dependence of residuals from fitted values (otherwise, the *weighted* least squares method have to be used; it will be discussed in the next paragraph 5.

Finally, the linear regression described above represents an appropriate prediction method when the dependent variable is a continuous variable (namely, it follows a Gaussian distribution). A generalization of linear regression can be made including non-continuous data for the dependent variable, and will be treated in the next paragraph 5.

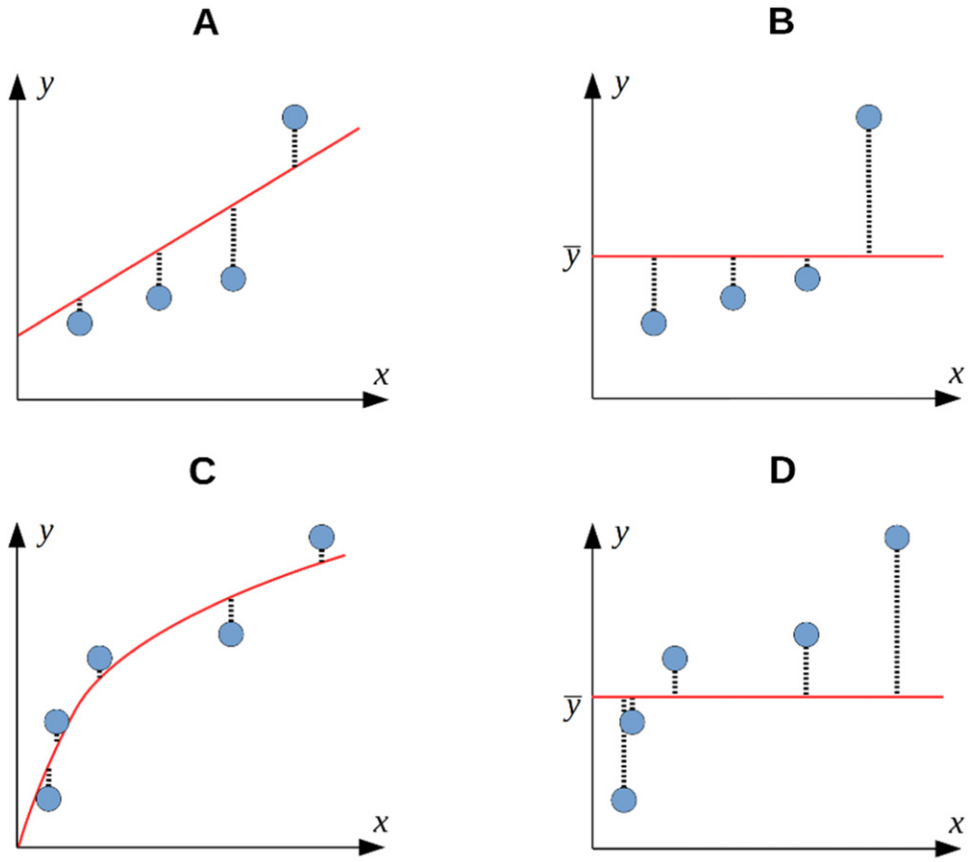

Fig. 7 Two cases of comparison between the fitting curve and the horizontal line passing through the mean value $\bar{y}$ of y_i data (B and D). Above: a case of linear regression (A); Below: a case of nonlinear regression (C).

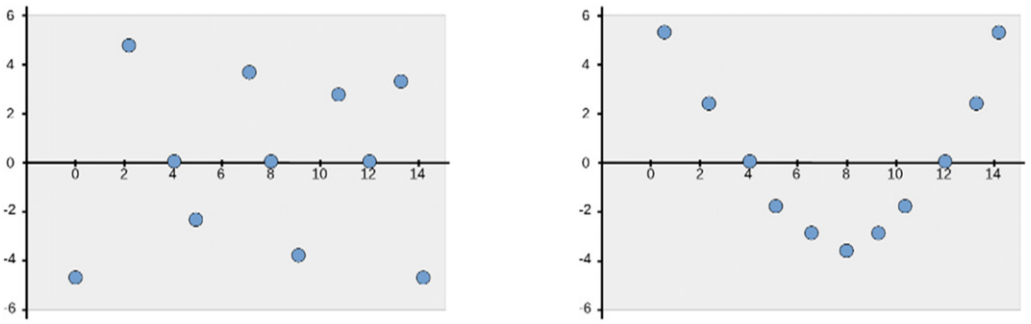

Fig. 8 Two typical patterns for residual plot: ϵon the y-axis vs a single independent variable x. a) the random plot suggest that the linear fitting model is a good choice to describe the data; b) a non-random plot suggests a nonlinear model.

Generalized Linear Models

The regression analysis is a statistical method used to estimate the possible relationship between two types of variables. According to the regression method, a dependent variable Y or response variable[8] is expressed as a function of one or more independent variables, also known as *predictor or explanatory variables* of the form $\boldsymbol{X} = (x_1, \ldots, x_n)$. Generalized Linear Model (GLM) (Nelder and

[8]Here, the response variable is indicated with uppercase to highlight that is a random variable.

Wedderburn, 1972) is a generalization of the linear regression. It allows to model categorical response variables as a function of explanatory variables by using linear regression.

Categorical response variables or observations can be binary, having "yes" or "no" possibilities as outcomes or more generally, "success" or "failure". To describe such binary observations of the response variable Y, it is usual to assume a binomial distribution for the data set y_i.

If the observations are counts, then a Poisson or negative binomial distribution is suitable to describe such data that will be nonnegative integers (Agresti, 1996).

Common types of GLMs are logistic regression and Poisson regression, which are employed respectively for the previous two cases. They will be described in the following sections.

Components of a Generalized Linear Model

A generalized linear model is composed of three components: i) random component, ii) systematic component, iii) link function. The *random component* specifies the response or dependent variable Y and the probability distribution hypothesized for it. The *systematic component* points out the explanatory or independent variables $(x_1, \ldots, x_n)$, which describe each instance $\boldsymbol{X}_i$ of the data set, where $i = 1, \ldots, p$ is the total number of instances in the data set. Values of the explanatory variables are treated as fixed and not as random variable. The *link function* $g(\mu)$ indicates a function of mean μ of the probability distribution of Y, being$\mu = E(Y)$ the expected value or mean of Y. The expected value μ of a probability distribution can change depending on the explanatory variables. For example, the probability of the incidence of a disease can be considered as function of the presence of a risk factor. GLMs uses a *prediction equation* or *model equation* to relate this expected value or mean to the explanatory variables through the link function. Such model equation has a linear form, as shown in Eq. (23):

$$g(\mu) = \alpha + \beta_1 x_1 + \ldots + \beta_n x_n \tag{23}$$

where the linear combination of the explanatory variables is known as *linear predictor*. α and β_j are the coefficient of regression, with $j = 1, \ldots, n$ if the independent variables are n.

The simplest link function is the *identity link* $g(\mu) = \mu$. If put in the left side of the Eq. (23), it describe a linear model for the mean response as a function of the independent variables. Using only one explanatory variable, the model equation with identity link has the form shown in the Eq. (24):

$$\mu(x) = \alpha + \beta x \tag{24}$$

This is known as ordinary linear regression model,[9] employed for continuous responses. Therefore, linear regression is a special case of GLMs, known as *Normal GLM* because it assume a normal distribution for Y.

Other link functions consider a nonlinear relation between μ and the predictor variables. An example is the *log link function*, which considers the log of the mean $g(\mu) = \log(\mu)$. Prediction Eq. (23) employs log link function as left side when the mean μ can not be negative, as with count data. GLM using the log link function is known as the *loglinear model*.

Another nonlinear link function is the *logit link:* $g(\mu) = \log\left(\frac{\mu}{1-\mu}\right)$, which models the log of an odds. It is convenient when $0 \leq \mu \leq 1$, as a probability. GLM using the logit link function is known as *logit model* or *logistic regression model*.

The next sections aim to describe GLMs for discrete responses in the two most important case: logistic regression models for binary data and loglinear models for count data.

Logistic Regression

Logistic regression (Cox, 1958) is the most common method for analysis of binary response data (Dobson, 2001). The logit model estimates the probability of a binary response Y (random component) as a function of one or more predictor or independent variables $\{x_j\}$ (systematic component), where $j = 1, \ldots, n$. For simplicity, it will be introduced the case of a single explanatory variable x.

The possible outcome for a response variable Y is denoted by 1 ("success") or 0 ("failure").

The distribution of the response variable Y is specified by the probabilities $P(Y = 1) = \pi$ or $P(Y = 0) = 1 - \pi$ for the single observation. The number y of successes in n independent observations follows the binomial distribution with index n and parameter π, as shown in the Eq. (25):

$$P(y;\pi) = \binom{n}{y} \pi^y (1-\pi)^y \tag{25}$$

where $y = 1, 2, \ldots, n$. The binomial distribution has expected value[10] or mean $E(Y) = \mu = n\pi$ and variance[11] $\text{var}(Y) = \sigma^2 = n\pi(1-\pi)$.

[9]The model Eq. (24) with regression coefficients α and β recalls the Eq. (7) of the linear regression method with regression coefficients respectively to w_0 and w_1.
[10]The expected value $E(Y)$ is the probability-weighted average of all possible outcomes of a random variable.
[11]The variance $var(Y) = \sigma^2 = E(Y\text{-}E(Y))^2$ is the expected value of the squared difference between a random variable and its mean.

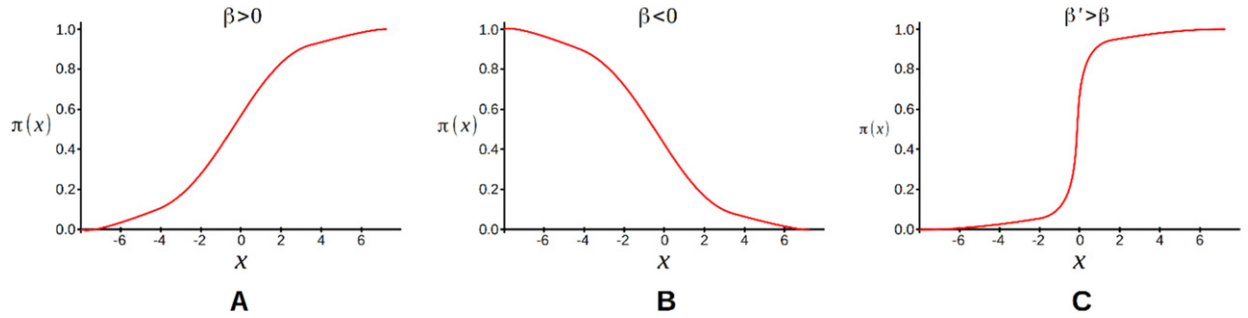

Fig. 9 Logistic regression functions, increasing and decreasing curve respectively for positives and negatives values of the β parameter. The second panel shows two different rate of change for two different positive value of β.

Binomial data are described by a GLM having the prediction Eq. (26):

$$g(\pi(x)) = \log\left(\frac{\pi(x)}{1-\pi(x)}\right) = \alpha + \beta x \tag{26}$$

where $\log\left(\frac{\pi(x)}{1-\pi(x)}\right)$ is the *logit function* of π, also known as "logit(π)". Logit function is the link function between the mean probability π and the linear regression expression. From the Eq. (26) it is possible to obtain the relationship between the success probability $\pi(x)$ and the predictor variable x. It is nonlinear, but varies continuously following a characteristic S-shaped curve, described by the so-called *logistic regression function* of Eq. (27):

$$\pi(x) = \frac{1}{1+e^{-(\alpha+\beta x)}} \tag{27}$$

where x is the single independent or explanatory variable, α and β are the coefficients of regression. The βvalue establish the rate of sigmoid variation. As shown in **Fig. 9**, the increase or decrease of the curve depends on the positives or negative value of β. The rate of change of the sigmoid becomes steeper for greater values of β. For $\beta = 0$ the binary response Y is independent of the explanatory variable x and the sigmoid becomes a horizontal line.

The outcomes of π are number between 0 and 1. They can be considered as the probability that the dependent variable Y is a success case or not (i.e. if the class label of Y is "yes" or "no" respectively).

Logistic regression classifier

Logistic regression can be used also to solve problems of classification. In general, logistic regression classifier can use a linear combination of more than one feature value or explanatory variable as argument of the sigmoid function. The corresponding output of the sigmoid function is a number between 0 and 1. The middle value is considered as threshold to establish what belong to the class 1 and to the class 0. In particular, an input producing an outcome greater than 0.5 is considered belong to the class 1. Conversely, if the output is less than 0.5, then the corresponding input is classified as belonging to 0 class (Harrington, 2012).

Nominal and ordinal logistic regression

Binomial or binary logistic regression is specific to treat a response variable with binary or binomial categories, as "true" or "false", "male" or "female", "healthy" or "sick" etc. Cases where the response variable considers more than two categories are described by nominal or ordinal logistic regression (Dobson, 2001), depending on whether they are ordered categories or not. An example of ordinal variable is one that can assume the values "low", "medium" or "high" as the level of response to a medical treatment. It is an ordinal variables because follow ordered scales, unlike the nominal variables, for example "classical", "rock", "blues" or "jazz" as favorite type of music. Methods used for nominal variables can be used for ordinal ones. Conversely it is not true (Agresti, 1996).

Poisson Regression

Poisson regression is the GLM used to describe frequencies or count data. Such data usually exhibit a Poisson distribution for the probability to count a number y of events in a given temporal interval, knowing the average number θ of events in that interval. Poisson distribution is shown in the Eq. (28):

$$P(y;\theta) = e^{-\theta}\frac{\theta^y}{y!} \tag{28}$$

where $y = 0, 1, 2, \ldots$ The only parameter of the distribution is θ and it can take positive values ($\theta > 0$); it expresses the mean and also the variance of the probability distribution, i.e. respectively $E(Y) = \theta$ and $\text{var}(Y) = \sigma^2 = \theta$. This means that the variability of the counts tends to increase when the number of counts increases.

The link function for this GLM is the log link function. Therefore, the prediction equation follows the loglinear model shown in Eq. (29):

$$g(\theta) = \log(\theta) = \alpha + \beta x \tag{29}$$

The mean θ is an exponential function of the explanatory variable x, as shown in the Eq. (30):

$$\theta(x) = \exp(\alpha + \beta x) = e^{\alpha}(e^{\beta})^{x} \tag{30}$$

The values of the parameter β determine how the mean θ varies with the explanatory variable, increasing or decreasing respectively for positives or negatives β. If $\beta = 0$ the means of Y is independent of x.

How GLMs Work

Estimating parameters: Maximum likelihood extimation

For a continuous random response variable Y the probability density function or probability distribution is denoted as $P(y;\boldsymbol{\phi})$, where y is the observation of the Y variable for a fixed $\boldsymbol{\phi}$, which is the vector of parameters of the distribution, with components like the expected value $E(Y)$ and variance var(Y). In general, the parameters values are unknown. Data sample allows estimate these parameters by using the so-called *likelihood function*. The likelihood function $L(\boldsymbol{\phi}; y)$ has the same algebraic form of the probability density function but conversely, L is a function of $\boldsymbol{\phi}$, for y fixed. The method of estimation of the parameter is based on the *Maximum Likelihood (ML) estimation*, which finds the parameter value that maximizes the likelihood function. The value of $\boldsymbol{\phi}$ which maximizes the likelihood function is called *maximum likelihood estimate of* $\boldsymbol{\phi}$, and is indicated as $\hat{\boldsymbol{\phi}}$. Frequently, it can be convenient to deal with the *log-likelihood function* $l(\boldsymbol{\phi}; y) = \log L(\boldsymbol{\phi}; y)$. Indeed, the parameter value sought $\hat{\boldsymbol{\phi}}$ will maximize also the log-likelihood function. It will be found by differentiating the $L(\boldsymbol{\phi}; y)$ or $l(\boldsymbol{\phi}; y)$ with respect to each ϕ_j, and solving the simultaneous equations $\frac{\partial L((\boldsymbol{\phi};y)}{\partial \phi_j} = 0$ for $j = 1, \ldots, p$. In practical applications, numerical methods allow find $\hat{\boldsymbol{\phi}}$ without solve these equations. GLMs works with such numerical approximations.

Example of application

Considering the binomial case, whose probability distribution is expressed by Eq. (25), likelihood function says how the probability of a given number of successes y in n trials varies as a function of the success parameter π of the single trial. It is expressed by $L(\pi; y)$, where $0 \le \pi \le 1$ for binomial distribution. For example, the success parameter for the heads output $(Y = 1)$ in a single throw of a fair coin is $\pi = 0.5$. The **Fig. 10** shows the likelihood function for two fixed numbers of success $y = 0$ and $y = 5$ in $n = 10$ throws. (The number of success could be considered as the head outcome in the coin throw). The likelihood function for $y = 0$ has the form $L(\pi; 0) = \binom{10}{0}\pi^0(1-\pi)^{10} = (1-\pi)^{10}$. For binomial outcomes of y successes in n trials, the likelihood function has a maximum for $\hat{\pi} = y/n$.

Estimating parameters: Least squares estimation

Another method of estimate for the parameter of a model is that of least squares, already introduced in the previous paragraph 4.2 (linear and nonlinear regression). Here, considering Y_i independent random variable with $i = 1, \ldots, p$ and indicating with μ_i their respective expected values, the hypothesis is that each μ_i is a function of a regression parameters vector $\boldsymbol{\beta}$. The estimator $\hat{\boldsymbol{\beta}}$ is that minimizes *RSS*, namely the sum of squares of the differences between observed Y_i and expected values $\mu_i(\boldsymbol{\beta})$: $RSS = \sum [Y_i - \mu_i(\boldsymbol{\beta})]^2$ (as seen above in Eq. (6)). $\hat{\boldsymbol{\beta}}$ is the parameter for which the *RSS* is minimized. Therefore, as seen in Eqs. (8) and (9), it is obtained by solving simultaneous equations of the type: $\frac{\partial RSS(\boldsymbol{\beta})}{\partial \beta_j} = 0$, where $j = 1, \ldots, n$.

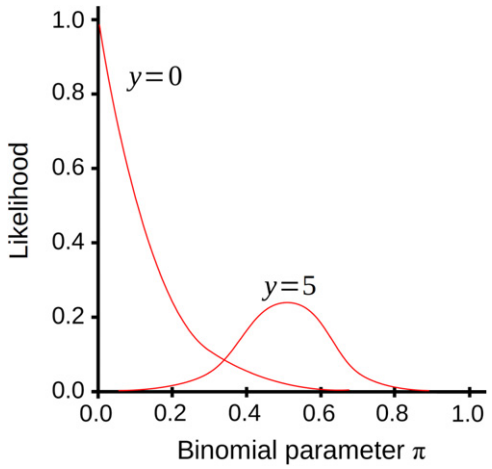

Fig. 10 Binomial likelihood functions for a number of success $y = 0$ and $y = 5$ successes in n = 10 trials. Likelihood function says that the probability of $y = 0$ for the head outcome in $n = 10$ throws has a maximum for $\hat{\pi} = 0$ (unfair coin).

If the variance ${\sigma_i}^2$ is not the same for each Y_i, then the weighted sum $RSS = \sum \frac{1}{{\sigma_i}^2}[Y_i - \mu_i(\boldsymbol{\beta})]^2$ is minimized, ensuring that the Y_i observations with greater variance, that mean less reliable, have a less effect on the estimates. This is the *weighted least squares method*.

Also for the least squares method, it is possible resort to numerical approach to find the estimator.

Under assumption of Gaussian distribution for the value of Y, least squares estimates coincide with the ML estimates (Agresti, 1996).

Fitting generalized linear models

Newton-Raphson algorithm

In GLMs, a numerical algorithm is used to find ML estimates as distribution parameters values $\hat{\boldsymbol{\phi}}$ of the model (as a function of regression coefficients $\hat{\boldsymbol{\beta}}$). It starts assuming an initial value for the parameters maximizing the likelihood function. Through successive approximations, the parameters values tend to get closer to the ML estimates. For binomial logistic regression model and Poisson loglinear regression model, the above-mentioned method is the *Newton-Raphson algorithm* (Agresti, 1996). This algorithm is a simplification of the Fisher scoring algorithm. Through the Newton-Raphson algorithm, the log-likelihood function is approximated by a second –degree polynomial curve in the neighborhood of the initial parameter guess. Indeed, for a such parabola-shaped function it is easier to determine the value corresponding to the maximum. This value will be the second guess for the ML estimate. Hence, the algorithm iterates the approximation with the concave parabola in the neighborhood of this second guess. This procedure is repeated until the location of the maximum does not change anymore.

Model checking

It is possible to evaluate the goodness of a fit considering the distance between the observed value y_i and the *fitted values* $\hat{\mu}_i$, namely the *residuals* $y_i - \hat{\mu}_i$, for each observation i of the dataset.

To check a model involving a Normal distribution (Normal GLM), it is usual to consider a *standardized residual* obtained by dividing it by the estimate $\hat{\sigma}$ of its unknown σ parameter ,[12] as shown in Eq. (31):

$$r_i = y_i - \hat{\mu}_i / \hat{\sigma} \tag{31}$$

If the assumption at the base of the model is correct, then the residuals should follow a Normal distribution with mean of zero and constant variance. They should be also independent from the explanatory variables. The goodness of the choice of the model adopted to describe the data is checked using suitable graphical methods. For example, a plot of residual against each explanatory variable should not show any pattern (as illustrate in **Fig. 8**). In addition, a plot of residuals against the fitted values $\hat{\mu}_i$ allows to detect an eventual change in variance, in violation of the assumption of constant variance (homoscedasticity) (as discussed in the section "How regression prediction methods work"). Further, to examine the goodness of the model, the residuals can be aggregated and, as already seen above (paragraph 4.1) it is preferable to consider the sum of the squares of residuals on the entire dataset: $\sum (y_i - \hat{\mu}_i)^2$

A quantitative test to check the goodness of a fitting model is the χ^2 in the following Eq. (32):

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i} \tag{32}$$

where the argument of the sum is the square of the standardized residuals of Eq. (31). In particular, the Eq. (32) becomes the Eq. (33) for the logistic model and the Eq. (34) for the Poisson:

$$\sum r^2 = \sum \frac{(y_i - n\hat{\pi}_i)^2}{n\hat{\pi}_i(1 - \hat{\pi}_i)} \tag{33}$$

$$\sum r^2 = \sum \frac{\left(y_i - \hat{\theta}_i\right)^2}{\hat{\theta}_i} \tag{34}$$

where the sum run over the total number of data considered.

The χ^2 test evaluates the agreement between the observed distribution of data and the expected GLM employed to describe the data. It provides a numeric value that allows establishing if the supposed GLM describes adequately the data.

Conclusive Remarks

GLMs represent a unifying model of several statistical methods. They are a generalization of the linear regression (section "How it Works") because include response variables that follow a distribution model different from the Gaussian model (therefore not necessarily continuous but also categorical response variable) (Agresti, 1996; Dobson, 2001).

[12]The standard error $\sigma = \sqrt{\text{var}(Y)}$, namely the root square of the variance.

Conclusions

The methods described in this chapter are aimed at regression, the prediction of the value corresponding to a new input; these methods are also indicated as regressors or function estimators.

The algorithms considered are K-nearest neighbor, genetic algorithms, linear and non-linear regression, among others. For each method we focused on their characteristics, highlighting advantages and critical issues.

Besides the theoretical considerations, we also report some practical considerations, helping the reader in their own implementations and developments of new applications.

References

Agresti, A., 1996. An Introduction to Categorical Data Analysis. John Wiley & Sons. p. 1996.

Aha, D., 1992. Tolerating noisy, irrelevant, and novel attributes in instance-based learning algorithms. International Journal of Man-Machine Studies 36, 267–287. (Elsevier).

Bhatia, N., 2010. Survey of nearest neighbor techniques. International Journal of Computer Science and Information Security 8 (2), 302–305.

Bishop, C.M., 2006. Pattern Recognition and Machine Learning. Singapore: Springer.

Boriah, S., Chandola, V., Kumar, V., 2008. Similarity measures for categorical data: A comparative evaluation. In: Proceedings of 2008 SIAM International Conference on Data Mining, pp. 243-254. Society for Industrial and Applied Mathematics.

Chang, E.I., Lippmann, R.P., 1991. Using genetic algorithms to improve pattern classification performance. Advances in Neural Information Processing Systems. 797–803.

Chun, H., Keleş, S., 2010. Sparse partial least squares regression for simultaneous dimension reduction and variable selection. The Journal of the Royal Statistical Society, Series B (Statistical Methodology) 72 (1), 3–25.

Cover, T.M., Hart, P.E., 1967. Nearest neighbor pattern classification. IEEE Transaction on Information Theory, Vol. IT-13, 21-27.

Cox, D.R., 1958. The regression analysis of binary sequences. Journal of the Royal Statistical Society, Series B (Methodological. 215–242.

DeJong, K.A., 1985. Genetic algorithms: A 10 year perspective. In: Proceedings of the First International Conference on Genetic Algorithms, pp. 169-177.

Dice, L.R., 1945. Measures of the amount of ecologic association between species. Ecology 26 (3), 297–302.

Dobson, A.J., 2001. An Introduction to Generalized Linear Models, second ed. Chapman and Hall.

Duda, R.O., Hart, P.E., Stork, D.G., 1991. Pattern Classification. New York: John Wiley & Sons.

Embrechts, M.J., Ekins, S., 2007. Classification of metabolites with kernel-partial least squares (K-PLS). Drug Metabolism and Disposition 35 (3), 325-7.

Fix, E., Hodges, J.L., 1952. Discriminatory analysis: Small sample performance. USAF School of Aviation Medicine, Randolph Field, Tex., Project 21-49-004, Rept. 11, August 1952.

Fix, E., Hodges Jr., J.L., 1951. Discriminatory analysis, nonparametric discrimination. USAF School of Aviation Medivine, Randolph Field, Tex., Project 21- 49-004, Rept. 4, Contract AF41(128)-31.

Flach, P., 2012. Machine Learning. New York: Cambridge University Press.

Gersho, A., Gray, R.M., 1992. Vector Quantization and Signal Compression. New York: Kluwer.

Ghahramani, Z.A., 2011. Tutorial on Gaussian processes (or why I don't use SVMs), In: Machine Learning Summer School (MLSS).

Goldberg, D., 1989. Genetic algorithms in search. Optimization, and Machine Learning. Boston, MA: Addison-Wesley.

Guo, G., Wang, H., Bell, D., Bi, Y., Greer, K., 2003. KNN model-based approach in classification. In: Meersman, R., Tari, Z., Schmidt, D.C. (Eds.) Proceedings of OTM Confederated International Conferences "On the Move to Meaningful Internet Systems", pp. 986-996.

Halstensen, M., Hundhausen, U., Esbensen, E.H., 2020. Partial least squares PLS1 vs. PLS2–optimal input/output modeling in a compound industrial drying oven. In: Proceedings of the 61st SIMS Conference on Simulation and Modelling SIMS, pp. 22-24.

Han, J., Kamber, M., Pei, J., 2012. Data mining, Concepts and Techniques, third ed. Waltham, MA: Morgan Kaufmann.

Harrington, P., 2012. Machine Learning in Action. Manning: Shelte Island (NY).

Hart, P.E., 1968. The condensed nearest neighbor rule. IEEE Transactions on Information Theory 14, 515–516.

Hastie, T., Tibshirani, R., Friedman, J., 2008. The Elements of Statistical Learning, second ed. Stanford, CA: Springer.

Hoerl, A.E., Kennard, R.W., 2000. Ridge regression: Biased estimation for nonorthogonal problems. Technometrics 42 (1), 80–86.

Holland, J.H., 1975. Adaptation in natural and artificial systems. Ann Arbor. MI: University of Michigan Press.

Janikow, C.Z., 1993. A knowledge-intensive genetic algorithm for supervised learning. Machine Learning 13 (2-3), 189–228.

Johnson, P., Vandewater, L., Wilson, W., *et al*., 2014. Genetic algorithm with logistic regression for prediction of progression to Alzheimer's disease. BMC Bioinformatics 15, 1–14.

Krasser, M., 2018. Gaussian Processes.

Levandowsky, M., Winter, D., 1971. Distance between sets. Nature 234 (5323), 34–35.

Michalewicz, Z., 1992. Genetic algorithms. Data Structures = Evolution Programs. USA: Springer Verlag.

Mitchell, M., 1996. An Introduction to Genetic Algorithms. Cambridge, MA: MIT Press.

Motulsky, H., Christopoulos, A., 2004. Fitting Models to Biological Data Using Linear and Nonlinear Regression: A Practical Guide to Curve Fitting. Oxford University Press.

Nelder, J., Wedderburn, R., 1972. Generalized Linear Models. Journal of the Royal Statistical Society 135, 370–384.

Preparata, F.P., Shamos, M.I., 1985. Computational Geometry: An Introduction. New York: Springer-Verlag.

Qian, G., Sural, S., Gu, Y., Pramanik, S., 2004. Similarity between Euclidean and cosine angle distance for nearest neighbor queries. In: Proceedings of the 2004 ACM Symposium on Applied Computing, pp. 1232-1237. ACM.

Rasmussen, C.E., Williams, C.K.I., 2006. Gaussian Processes for Machine Learning. The MIT Press.

Rechenberg, I., 1973. Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution, Frommann-Holzboog Verlag: Stuttgart.

Riolo, R.L., 1992. Survival of the fittest bits. Scientific American 267 (1), 114–116

Rojas, R., 1996. Neural Networks. A systematic Introduction. Berlin: Springer.

Ruckstuhl, A., 2010. Introduction to Nonlinear Regression.

Russolillo, G., 2012. Non-metric partial least squares. Electronic Journal of Statistics 6.

Sivanandam, S.N., Deepa, S.N., 2007. Introduction to genetic algorithms. Heidelberg: Springer Science & Business Media.

Stanfill, C., Waltz, D., 1986. Toward memory-based reasoning. Communications of the ACM 29 (12), 1213–1228.

Stock, J.H., Watson, M.W., 2003. Introduction to Econometrics, vol. 104. Boston: Addison Wesley.

Tanimoto, T.T., 1958. Elementary Mathematical Theory of Classification and Prediction.

Tenenhaus, M., 1998. La Regression PLS: theorie et pratique, Technip, Paris.

Tenenhaus, M., Esposito Vinzi, V., Chatelin, Y., Lauro, C., 2005. PLS path modeling. Computational Statistics and Data Analysis 48, 159–205.

Tibshirani, R., 1996. Regression shrinkage and selection via the Lasso. Journal of the Royal Statistical Society. Series B (Methodological) 58 (1), 267–288.
Trygg, J., Wold, S., 2002. Orthogonal projections to latent structures. Journal of Chemometrics 16 (3), 119–128.
Wu, T.T., Chen, Y.F., Hastie, T., Sobel, E., Lange, K., 2009. Genome-wide association analysis by lasso penalized logistic regression. Bioinformatics 25 (6), 714–721.
Zou, H., Hastie, T., 2005. Regularization and Variable Selection via the Elastic Net. Journal of the Royal Statistical Society. Series B (Statistical Methodology) 67 (2), 301–320.

Further Readings

Aamodt, A., Plaza, E., 1994. Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI communications 7 (1), 39–59.
Allen, B.P., 1994. Case-based reasoning: Business applications. Communications of the ACM 37, 40–42.
Agresti, A., 1990. Categorical Data Analysis. New York: Wiley.
Bailey, T., Jain, A.K., 1978. A note on distance-weighted k-nearest neighbor rules. IEEE Transactions on Systems, Man, and Cybernetics 4, 311–313.
Booker, L., B., Goldberg, D.E., Holland, J.H., 1989. Classifier systems and genetic algorithms. Artificial Intelligence 40, 235–282.
Dasarathy, B., 1991. Nearest Neighbor Pattern Classification Techniques. Silver Spring, MD: IEEE Computer Society Press.
Davis, L., 1991. Handbook of Genetic Algorithms. New York: Van Nostrand Reinhold.
Finnie, G.R., Wittig, G.E., Desharnais, J.M., 1997. A comparison of software effort estimation techniques: Using function points with neural networks, case-based reasoning and regression models. Journal of Systems and Software 39 (3), 281–289.
Friedman, J.H., 1991. Multivariate adaptive regression splines. The Annals of Statistics 19 (1), 67.
Harrell, F., 2015. Regression Modeling Strategies: With Applications to Linear Models, Logistic and Ordinal Regression, and Survival Analysis. Springer.
Kim, G.H., An, S.H., Kang, K.I., 2004. Comparison of construction cost estimating models based on regression analysis, neural networks, and case-based reasoning. Building and Environment 39 (10), 1235–1242.
Kolodner, J., 1983. Maintaining organization in a dynamic long-term memory. Cognitive Science 7, 243–280.
Kolodner, J.L., 1993. Case-Based Reasoning. San Matteo, CA: Morgan Kaufmann.
Koton, P., 1988. Reasoning about evidence in causal explanation.In: Proceedings of 7th National Conference of Artificial Intelligence (AAAI'88), pp 256–263.
Leake, D.B., 1996. CBR in context: The present and future. In: Leake, D.B. (Ed.), Case-Based Reasoning: Experiences, Lessons, and Future Directions. Menlo Park: AAAI Press, pp. 3–30.
Riesbeck, C., Schank, R., 1989. Inside Case-Based Reasoning. Hillsdale: Lawrence Erlbaum.
Rissland, E.L., Ashley, K., 1987. HYPO: A case-based system for trade secret law.In: Proceedings of 1st International Conference on Artificial Intelligence and Law, pp 60–66. Boston, MA: ACM.
Schank, R.C., 1982. Dynamic Memory: A Theory of Reminding and Learning in Computers and People. Cambridge, UK: Cambridge University Press.
Schank, R.C., 1984. Memory-based expert systems. Technical Report (# AFOSR. TR. 84-0814). New Haven, USA: Yale University.
Whitley, D., 1994. A genetic algorithm tutorial. Statistics and Computing 4 (2), 65–85.