Chapter 2
Machine Learning

Machine learning is currently a mainstream research hotspot in the AI industry, entailing multiple disciplines such as probability theory, statistics, and convex optimization. This chapter first introduces the definition of "learning" in learning algorithms and the process of machine learning. On this basis, it offers some commonly used machine learning algorithms. Our readers will learn about some key concepts such as hyperparameters, gradient descent, and cross-validation.

2.1 Introduction to Machine Learning

Machine learning (including its branch deep learning) is the study of "learning algorithms". The so-called "learning" here refers to the situation that the performance of a computer program measured by performance metric P on a certain task T improves itself with experience E, then we call this computer program learning from experience E. For instance, identifying junk email is a task T. We can complete such tasks easily, because we have lots of experiences in doing so in daily life. These experiences may come from daily emails, spam messages or even advertisements on TV. We can summarize and conclude from these experiences that emails from unknown users that contain the words like "discount" and "zero risk" are more likely to be spam. Referring to the learnt knowledge, we can distinguish whether an email that has never been read before is spam, as shown in Fig. 2.1a. So, can we write a computer program to simulate the above process? As shown in Fig. 2.1b, we can prepare a number of e-mails, and pick out the junk emails by hands, as the experience E for this program. However, the program cannot automatically summarize these experiences. At this time, it is necessary to train the program through machine learning algorithms. The computer program that has been trained is called a model, and in general the larger the number of emails is used for training, the better the model may be trained, and the larger the value of the performance metric P will be.

Huawei Technologies Co., Ltd., *Artificial Intelligence Technology*,
https://doi.org/10.1007/978-981-19-2879-6_2

Fig. 2.1 Learning mode

Spam identification is very difficult to achieve through traditional programming methods. Theoretically, we should be able to find a set of rules in line with the features of all junk emails, rather than that of the regular emails. This method of using explicit programming to solve problems is known as a rule-based approach. However, in practice, it is almost impossible to find such a set of rules. Therefore, to solve this problem, machine learning adopts the statistical-based method. We can basically claim that machine learning is a method to let machines to learn rules automatically through samples. Compared with using the rule-based method, machine learning can learn more complex rules and the rules that are difficult to describe, so as to handle more complicated tasks.

Machine learning is highly adaptable and can solve many problems in the AI field, but this does not mean that machine learning will always be used preliminary in all cases. As shown in Fig. 2.2, machine learning is suitable for problems which require a complex solution or involve a large amount of data, but the probability distribution of the data is unknown. Machine learning can certainly solve the problem in other cases, but the cost is often higher than traditional methods. Take the second quadrant shown in Fig. 2.2 as an example. If the size of the problem is small enough to be solved by artificial rules, then there is no need to use machine learning algorithms. There are two main application scenarios in general for machine learning.

1. The rules are quite complicated or unable to be described, such as face recognition and voice recognition.
2. The data distribution itself changes over time, and the program needs to be constantly re-adapted, such as predicting the trend of commodity sales.

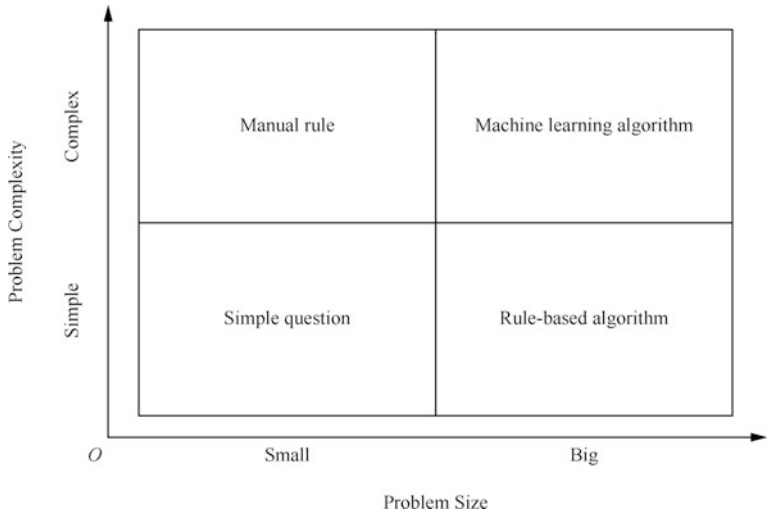


Fig. 2.2 Application scenarios of machine learning

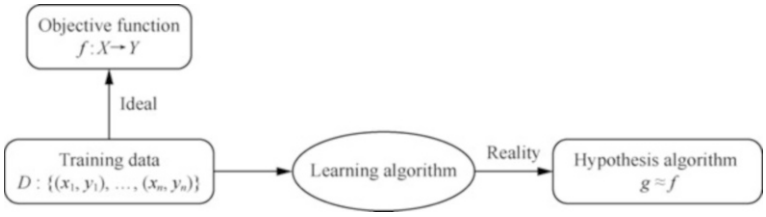


Fig. 2.3 The relationship between the hypothesis function and the objective function

2.1.1 Rational Understanding of Machine Learning Algorithms

The nature of machine learning algorithms is function fitting. Let f be the objective function, the purpose of the machine learning algorithm is to give a hypothetical function g that makes $g(x)$ and $f(x)$ as close as possible to the input x in any defined domain. A simple example is the probability density estimation in statistics. By the law of large numbers, the height of all Chinese population should follow a normal distribution. Although the probability density function f of this normal distribution is unknown, we can estimate the mean and variance of the distribution by sampling, and then estimate f.

The relationship between the hypothesis function and the objective function is shown in Fig. 2.3. For a given task, we can collect a large amount of training data. These data must fit a given objective function f, otherwise, it is meaningless to learn such a task. Next, the learning algorithm can analyze these training data and give a hypothetical function g that is similar to the objective function f as much as possible.

Therefore, the output of the learning algorithm is always not perfect and cannot be completely consistent with the objective function. However, with the expansion of the training data, the degree of approximation of the hypothesis function g to the objective function f is gradually improved, and finally a satisfactory level of accuracy can be achieved in machine learning.

It is worth mentioning that the objective function f is sometimes very abstract in existence. For the classic image classification task, the objective function means the mapping from the image set to the category set. In order to let a program to process logical information such as images and classes, it is necessary to use certain encoding methods to map images or categories one by one into scalars, vectors or matrices. For example, you can number each category from 0, thereby mapping the class to a scalar. You can also use different one-hot vectors to represent different classes, which is called one-hot encoding. The encoding of image is a little more complicated, and is generally represented by a three-dimensional matrix. With this encoding method, we can see the domain of definition the objective function f as a collection of three-dimensional matrices and its range as a collection of a series of serial numbers for classes. Although the encoding process is not part of the learning algorithm, in some cases, the choice of encoding method will also affect the efficiency of the learning algorithm.

2.1.2 Major Problems Solved by Machine Learning

Machine learning can deal with various types of problems, including the most typical ones such as classification, regression, and clustering. Classification and regression are the two major types of prediction problems, taking up of 80–90% of all the problems. The main difference is that the output of classification is discrete serial numbers of classes (generally called as "labels" in machine learning), while the output of regression is continuous value.

The classification problem requires the program to indicate which of the k classes does the input belong to. To solve this problem, machine learning algorithms usually output mapping from domain D to category labels $\{1, 2, \ldots, k\}$. Image classification task is a typical classification problem.

In regression problems, the program needs to predict the output value for a given input. The output of a machine learning algorithm is usually a mapping from the domain D to the real number domain R. For instance, predicting the claim amount of the insured (used to set insurance premium), or predicting the price of securities in the future are all relevant cases. In fact, classification problems can also be reduced to regression problems. By predicting the probability of the image belonging to each class, the machine learning can obtain the result of the classification.

The clustering problem needs to divide the data into multiple categories according to the inherent similarity of the data. Unlike the classification problem, the dataset of the clustering problem does not contain manually labeled labels. The clustering algorithm makes the data similar to each other within the class as much as

possible, while the data similarity between the classes is relatively small, so as to implement classification. Clustering algorithms can be used in scenarios like image retrieval, user portrait generation and etc.

2.2 Types of Machine Learning

According to whether the training dataset contains manually tagged labels, machine learning can be generally divided into two types—supervised learning and unsupervised learning. Sometimes, in order to distinguish it from unsupervised learning, supervised learning is also called learning under supervision. If some data in the dataset contains labels and the majority of the data does not, then this learning algorithm is called semi-supervised learning. Reinforcement learning mainly focuses on multi-step decision-making problems, and automatically collects data for learning in the interaction with the environment.

2.2.1 Supervised Learning

Generally speaking, supervised learning is allowing the computer to compare standard answers when it is trained to answer the multiple-choice questions. The computer tries its best to adjust its model parameters, expecting that the inferred answer is as consistent as possible with the standard answer, and finally learn how to solve the question. Using samples of known labels, supervised learning can train an optimal model meeting the required performance. Using this model, any input can be mapped to the corresponding output, so as to predict the unknown data.

Figure 2.4 shows a supervised learning algorithm that is highly simplified. The features in the graph can be understood as data items. Although this interpretation is not sufficient to some extent, it will not affect our description on this supervised learning algorithm. The supervised learning algorithm takes features as input and the predicted value of the targets as output. Figure 2.5 shows a practical example. In this example, we hope to make an overall prediction that whether a user enjoys exercises

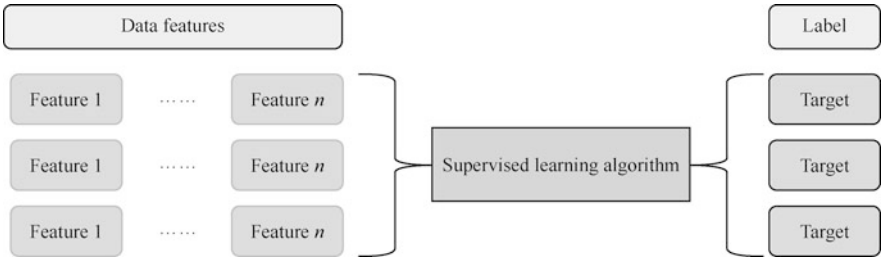


Fig. 2.4 Supervised learning algorithm

Fig. 2.5 Sample data

Feature			Target
Weather	Temperature	Wind speed	Enjoyment of exercise
Sunny	Warm	Strong	Yes
Rainy	Cold	Medium	No
Sunny	Cold	Weak	Yes

by taking weather conditions into account. Each row in the table is a set of training examples that records the weather characteristics of a specific day and the user's enjoyment of exercises. Similar algorithms can be applied to other scenarios such as product recommendations.

The input (feature) and output (target) of a supervised learning algorithm can be either continuous or discrete. When the value of the target variable is continuous, the output of the supervised learning algorithm is called a regression model. The regression model reflects the features of the attribute values in the sample dataset, and expresses the relationship of the sample mapping through functions to show the dependency between the attribute values. The attribute value mentioned here includes feature and target. Regression model is widely used in time series forecasting. For instance, how much money can you earn form stocks next week? What will the temperature be tomorrow? So on and so forth. Correspondingly, when the target variable takes discrete values, the output of the supervised learning algorithm is called a classification model. Through the classification model, the samples in the sample dataset can be mapped to the given classes. Such as whether there will be traffic jams on a highway during the morning rush hour tomorrow? Which one is more attractive to customers, a five-yuan voucher or a 25%-off discount? Etc.

Although the range of the regression model can be an infinite set, the output of the classification model is often finite. This is because the size of the dataset cannot grow indefinitely, and the number of classes in the dataset should to the most be the same with the number of training examples. Therefore, the number of classes is not infinite. When training a classification model, an artificially designated class set L is often needed for the model to select category output. The size of the set L is generally recorded as K, which is the number of possible classes.

2.2.2 Unsupervised Learning

Compared with supervised learning, unsupervised learning is like letting the computer do multiple-choice questions without telling it what the correct answer is. In this case, it is difficult for the computer to secure the correct answer. But by analyzing the relationship between these questions, the computer can classify the questions so that the multiple-choice questions in each class have the same answer.

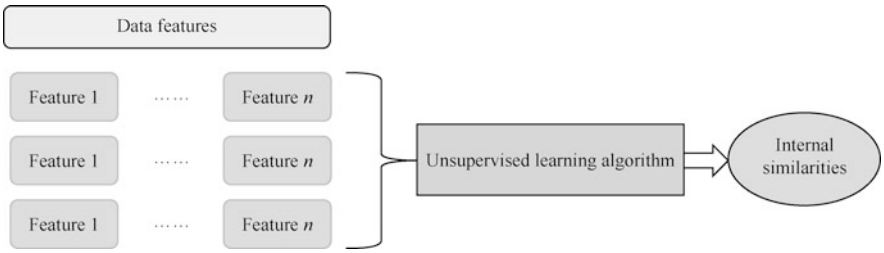


Fig. 2.6 Unsupervised learning algorithm

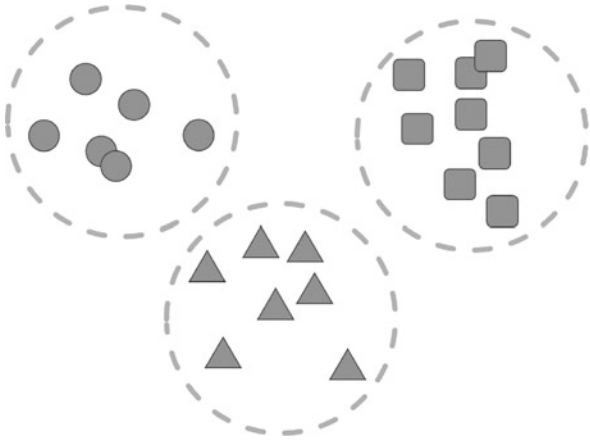


Fig. 2.7 Example of clustering algorithm

The unsupervised learning algorithm does not require labeling samples, but directly modeling the input dataset, as shown in Fig. 2.6.

Clustering algorithm is a typical unsupervised learning algorithm that can be summarized by the proverb: "birds of a feather flock together". The algorithm only puts the samples of high similarity together. For newly input samples, it only needs to calculate their similarity with the existing samples, and then classify them according to the degree of similarity. Biologists have used the concept of clustering to study the interspecies relationship for a long time. As shown in Fig. 2.7, after classifying the iris flowers based on the size of sepal and petal, the iris has been divided into three categories. Through the clustering model, the samples in the sample dataset can be divided into several categories, making the similarity of samples under the same category relatively higher. The scenarios of application for the clustering model include that what kinds of audiences like to watch movies of the same subject, and which components are damaged similarly, etc.

2.2.3 *Semi-supervised Learning*

Semi-supervised learning is a combination of supervised learning and unsupervised learning. This algorithm attempts to allow the learner to automatically utilize a large amount of unlabeled data to assist the learning of a small amount of labeled data. Traditional supervised learning algorithms need to learn from a large number of labeled training samples to build a model for predicting the labels of new samples. For example, in a classification task, the label suggests the class of the sample. And in a regression task, the label suggests the real-valued output of the sample. With the rapid development of human's ability to collect and store data, in many practical tasks, it is very easy to acquire a large amount of unlabeled data, and labeling them often consumes a lot of efforts and materials. For example, for webpage recommendation, users are required to mark web pages they interest in. But few users are willing to spend a lot of time to mark, so the web pages with marked information are limited. But there are countless web pages that are not marked, or marked, which can be used as unmarked data.

As shown in Fig. 2.8, semi-supervised learning does not require manual labeling on all the samples like supervised learning, nor is it completely independent from the target like unsupervised learning. In the semi-supervised learning datasets, generally speaking, there are only a few samples labeled. Taking the iris classification problem presented in Fig. 2.7 as an example. A small amount of supervised information is added to the dataset, as shown in Fig. 2.9. Let's assume the circle represents the Setosa sample, the triangle represents the Versicolor sample, the square represents the Virginica sample, and the star represents the unknown sample. The clustering algorithm has been introduced in unsupervised learning, suppose its output is shown by the dotted circle in Fig. 2.9. Counting the circles including the highest number of known samples and then this class can be used as the class for this cluster. To be more specific, the upper-left cluster belongs to Setosa, while the upper right cluster obviously belongs to Virginica. By combining unsupervised algorithm and supervised information, semi-supervised algorithm can achieve higher accuracy with lower labor costs.

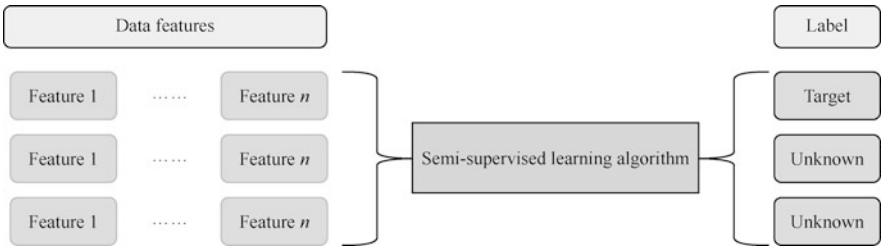


Fig. 2.8 Semi-supervised learning algorithm

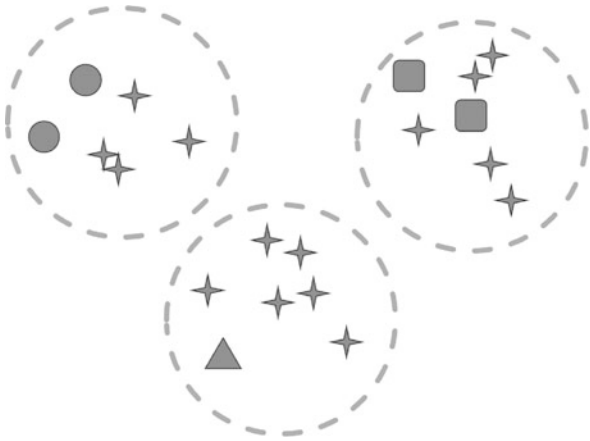


Fig. 2.9 Iris flower dataset with supervised information

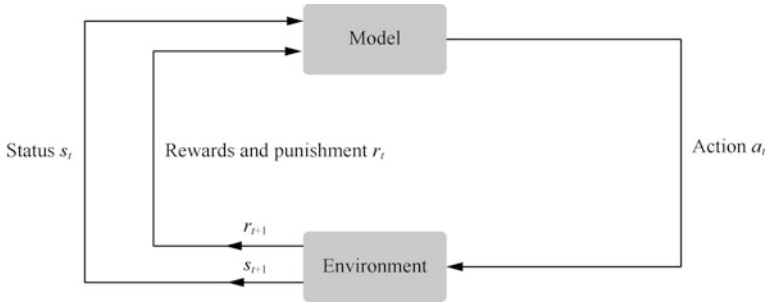


Fig. 2.10 Reinforcement learning algorithm

2.2.4 Reinforcement Learning

Reinforcement learning is mainly used to solve multi-step decision-making problems, such as Go game, video games, and visual navigation. Unlike the problems studied by supervised learning and unsupervised learning, these problems are often difficult to find accurate answers. Taking Go as an example. It takes about 10,170 operations to exhaust the results of the game (there are only 1080 atoms in the universe). So, for a given and common situation, it is difficult to find the perfect move.

Another characteristic of the multi-step decision problem is that it is easy to define a reward function to evaluate whether the task has been completed. The reward function of Go can be defined as whether to win the game; the reward function of electronic games can be defined as the score. The goal of reinforcement learning is to find an action strategy to maximize the value of the reward function.

As shown in Fig. 2.10, the two most important parts of a reinforcement learning algorithm are the model and the environment. In different environments, the model can determine its own actions, and different actions may have different effects on the

environment. Still, in the case of solving test questions, the computer can give the answer randomly, and the teacher will give a score based on the answer given. But if the situation is only limited to this case, it is impossible for the computer to learn how to solve the question, because the teacher's grading does not contribute to the training process. In this case, the importance of status and rewards and punishments are highlighted. A higher test score can make the teacher satisfied and then give the computer a certain reward. On the contrary, a lower test score may incur penalties. As a "motivated" computer, it is bound to hope that by adjusting its own model parameters, it can get more rewards because of its answers. In this process, no one provides training data for the learning algorithm or tells the reinforcement learning system how to make the correct move. All data and reward signals are dynamically generated during the interaction between the model and the environment and are automatically and dynamically learned. No matter it is good behavior or bad behavior, it can help the model to learn.

2.3 The Overall Process of Machine Learning

A complete machine learning project often involves data collection, data cleaning, feature extraction and selection, model training, model evaluation and testing, model deployment and integration, as shown in Fig. 2.11. This section introduces the concepts related to data collection and data cleaning, which are fundamental to understand what is feature selection. After selecting reasonable features, it is necessary to train and evaluate the model based on these features. This is not a one-kick process, but requires constant feedback and iteration to harvest satisfactory results. At last, the model needs to be deployed to the specific application scenarios to put theories into practice.

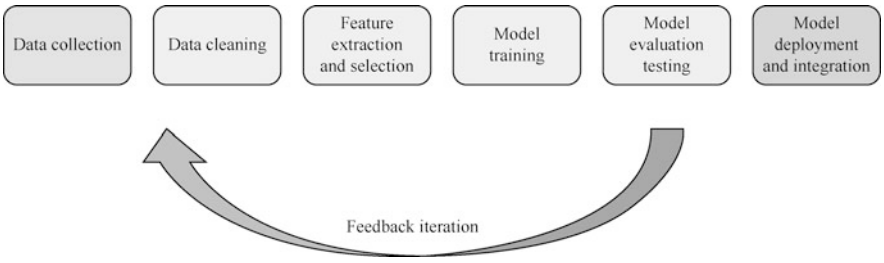


Fig. 2.11 The overall process of machine learning

2.3.1 Data Collection

A dataset is a set of data used in a machine learning project, and each data is called a sample. The items or attributes that reflect the performance or nature of the sample in a certain aspect are called features. The dataset used in the training process is called the training set, and each sample is called a training sample. Learning (training) is the process of learning a model from data. The process of using the model to make predictions is called testing, and the dataset used for testing is known as the test set. Each sample in the test set is called a test sample.

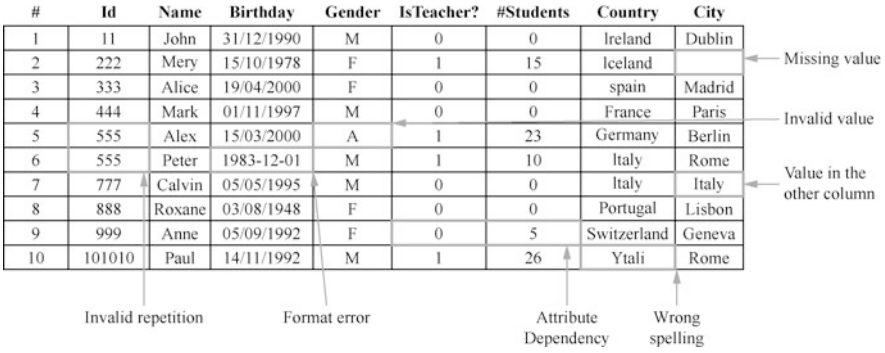
Figure 2.12 shows a typical dataset. In this dataset, each row indicates a sample, and each column refers to a feature or label. When the task is determined, such as predicting housing prices based on floor area, school district, and house orientation, the features and labels are also determined. Therefore, the row and column headers of the dataset should remain unchanged throughout the machine learning project. The training set and the test set are relatively free to split, and the researcher can determine which samples belong to the training set based on experience. A too low proportion of the test set may result in randomness of model testing, not able to properly evaluate the performance of the model. While a high proportion of the training set may result in low sample utilization and the model cannot learn thoroughly. Therefore, the common ratio between training set and dataset is that the training set accounts for 80% of the total number of samples, and the test set accounts for 20%. In this example, there are four samples in the training set and one sample in the test set.

2.3.2 Data Cleaning

Data is vital to the model and determines the limit of the model's capabilities. Without good data, there will be no good models. However, data quality is a

		Feature 1	Feature 2	Feature 3	Label
	Serial number	Floor area	School district	Orientation	Housing price
Training set	1	100	8	South	1000
	2	120	9	Southwest	1300
	3	60	6	North	700
	4	80	9	Southeast	1100
Test set	5	95	3	South	850

Fig. 2.12 Sample dataset



#	Id	Name	Birthday	Gender	IsTeacher?	#Students	Country	City
1	11	John	31/12/1990	M	0	0	Ireland	Dublin
2	222	Mery	15/10/1978	F	1	15	Iceland	
3	333	Alice	19/04/2000	F	0	0	spain	Madrid
4	444	Mark	01/11/1997	M	0	0	France	Paris
5	555	Alex	15/03/2000	A	1	23	Germany	Berlin
6	555	Peter	1983-12-01	M	1	10	Italy	Rome
7	777	Calvin	05/05/1995	M	0	0	Italy	Italy
8	888	Roxane	03/08/1948	F	0	0	Portugal	Lisbon
9	999	Anne	05/09/1992	F	0	5	Switzerland	Geneva
10	101010	Paul	14/11/1992	M	1	26	Ytali	Rome

Fig. 2.13 "Dirty" data

commonly problem bothering real data, as shown in Fig. 2.13. Following are some typical problems on data quality.

1. Incomplete: Data lacks attributes or containing missing values.
2. Noisy: Data contains erroneous records or outliers.
3. Inconsistent: Data contains conflicting records or discrepancies.

Such data is called "dirty" data. The process of filling in missing values, finding and eliminating data abnormalities is called data cleaning. In addition, data preprocessing often involves data dimensionality reduction and data standardization. The purpose of data dimensionality reduction is to simplify data attributes and avoid dimensional explosion; while the purpose of data standardization is to unify the dimensions of each feature, thereby reducing the difficulty of training. The content of data dimensionality reduction and data standardization will be introduced in detail in the later passages, and this section only talks about data cleaning.

What are handled by the machine learning model are all features. The so-called feature is the numerical representation of the input variable that can be used in the model. In most cases, the collected data can be used by the algorithm after preprocessing. The preprocessing operation mainly includes the following procedures.

1. Data filtering.
2. Handling missing data.
3. Handling possible errors or outliers.
4. Combining data from multiple sources.
5. Data aggregation.

The workload of data cleaning is often quite heavy. Research shows that data scientists spend 60% of their time on cleaning and organizing data in machine learning researches, as shown in Fig. 2.14. On the one hand, this shows how difficult data cleaning is, and that the data collection featuring different methods and contents will require different methods for data cleaning. What is more, it also shows that data cleaning plays a crucial role in subsequent model training and optimization. Another

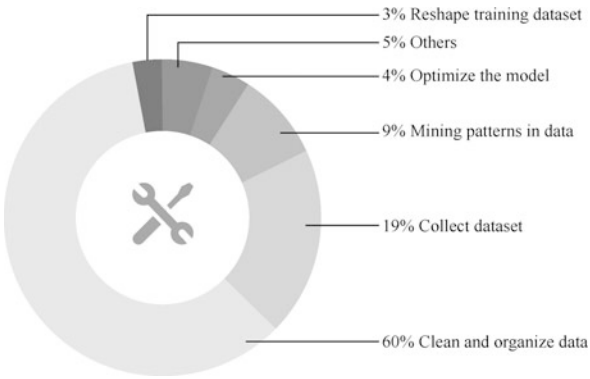


Fig. 2.14 The importance of data cleaning

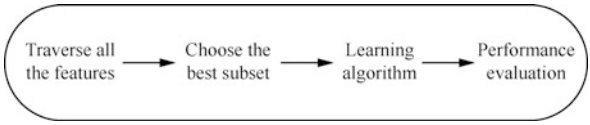


Fig. 2.15 The machine learning process using the filtering

important role is that the more thoroughly the data is cleaned, the less likely the model is to be affected by abnormal data, thus ensuring the model's training performance.

2.3.3 Feature Selection

Usually, there are many different features in a dataset, some of which may be redundant or unrelated to the target. For example, when predicting housing prices based on floor area, school district, and daily temperature, the daily temperature is apparently an irrelevant feature. Through feature selection, these redundant or irrelevant features can be eliminated, so that the model is simplified and easier to be interpreted by users. At the same time, feature selection can also effectively reduce the time of model training, avoid dimensional explosion, improve the generalization performance of the model, and avoid overfitting. Common methods for feature selection include filter methods, wrapper methods, and embedded methods, which will be introduced successively in the following passages.

The filter method is independent when selecting features and has nothing to do with the model itself. By measuring the correlation between each feature and the target attribute, filter method applies a statistical measurement to score each feature. By sorting these features on the basis of the scores, you can decide to keep or eliminate specific features. Figure 2.15 shows the machine learning process using filter methods. Statistical measures commonly used in filtering include Pearson's correlation coefficient, Chi-Square coefficient, and mutual information. Since filter

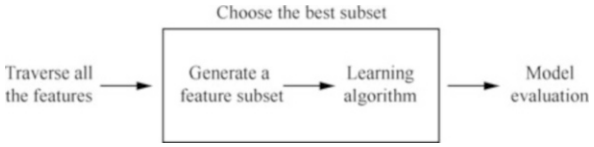


Fig. 2.16 The machine learning process using wrappers

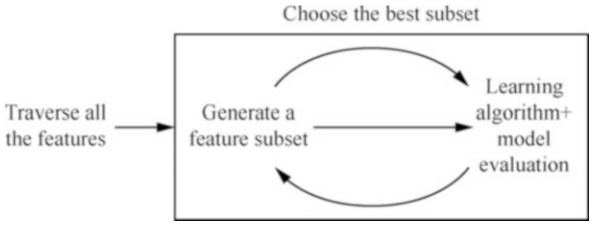


Fig. 2.17 The machine learning process using embedded method

does not consider the relationship between features, it is only prone to select redundant variables.

The wrapper method uses a predictive model to score a subset of features, and considers the feature selection problem as a search problem, where the wrapper will evaluate and compare different feature combinations, and the predictive model will be used as a tool for evaluating feature combinations. The higher the accuracy of the prediction model, the more the feature combination should be retained. Figure 2.16 displays the machine learning using the wrapper method. One of the popular wrapper methods is recursive feature elimination (RFE). Wrapper methods usually provide the best-performing feature set for a specific class of model, but it needs to train a new model for each feature subset, so the amount of operations is extensive.

The embedded method uses feature selection as part of model building, as shown in Fig. 2.17. Unlike the filter and wrapper method, the model using the embedded method actively learns how to perform feature selection during training. The most common embedded feature selection method is regularization. Regularization is also called the penalty method. By introducing additional constraints when optimizing the prediction algorithm, the complexity of the model is reduced, namely, the number of features is reduced. Common regularization methods include ridge regression and Lasso regression.

2.3.4 The Construction of Machine Learning Models

After finishing data cleaning and feature extraction, it is time to build the model. Taking supervised learning as an example, model construction generally follows the steps shown in Fig. 2.18. The core of model construction is model training, verification and testing. This section briefly explains the training and prediction process using one simple example. More details will be introduced in the following chapters.

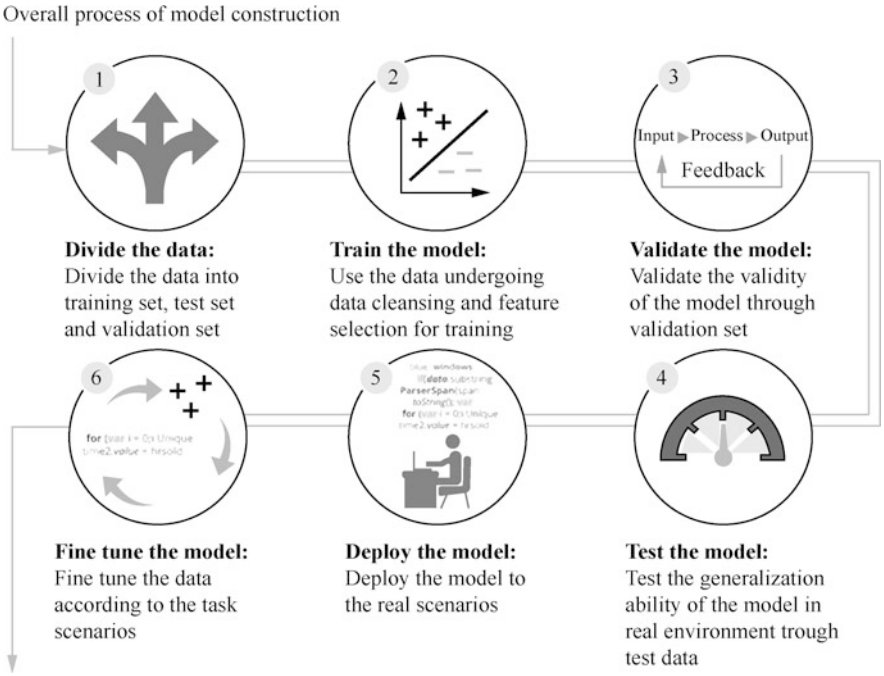


Fig. 2.18 Overall process of model construction

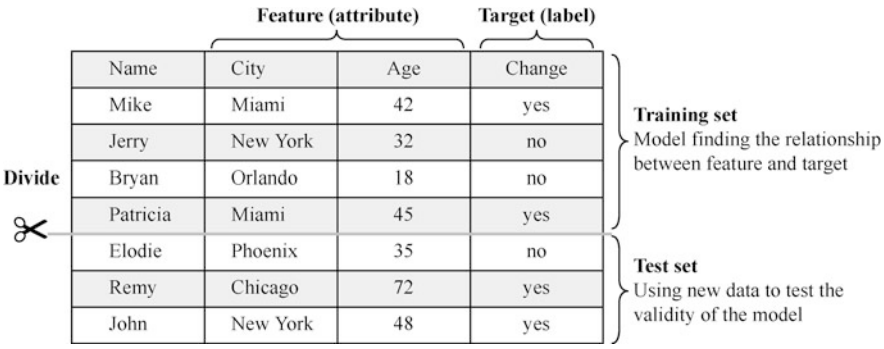


Fig. 2.19 Training set and test set

In the example of this section, we need to use a classification model to determine whether someone needs to change suppliers under certain features. Assuming that Fig. 2.19 shows the cleaned dataset, the task of the model is to predict the target as accurately as possible on the basis of the known features. During the training process, the model can learn the mapping relationship between features and targets based on the samples in the training set. After training, we can get the following model:

```
def model(city, age):
 if city == "Miami": return 0.7
 if city == "Orlando": return 0.2
 if age > 42: return 0.05 * age + 0.06
 else: return 0.01 * age + 0.02
```

The output of the model is the probability of truth value of the target. We know that as the training data increases, the accuracy of the model will also increase accordingly. So why not use all the data for training, instead of only taking a part of it as the test set? This is because that the performance of the model in the face of unknown data, not the known data, is what we should look at. The training set is like an exam bank that students read through while preparing for an examination. It is not a surprising thing no matter how high the accuracy rate will be for the students, because the exam bank always has a limitation. As long as they have a good memory, the students can even memorize all the answers after all. Only the formal examination can really test the students' acquisition of knowledge, because the questions in the official examinations may be something they have never seen before. The test set is equivalent to an examination prepared by the researchers for the model. In other words, in the entire dataset (including training set and test set), the model has the right to consult only the features of the training set and test set. The target of the test set can only be used by the researchers when evaluating the performance of the model.

2.3.5 Model Evaluation

What is a good model? The most important evaluation indicator is the model's generalization ability, also known as the prediction accuracy of the model dealing with actual business data. There are also some engineering indicators that can be used to evaluate the model: interpretability, which describes the degree of straightforwardness of the model's prediction results; prediction rate, which refers to the average time it takes for the model to predict each sample; plasticity, which refers to the acceptability of model prediction rate in actual business process as the business volume expands.

The goal of machine learning is to make the learned model applicable to new samples, not just on training samples. The ability of the learned model to apply to new samples is called generalization ability, also addressed as robustness. The difference between the predicted result of the learned model on the sample and the true result of the sample is called error. The training error refers to the error of the model on the training set, and the generalization error refers to the error of the model on the new sample (test set). Obviously, we want to have a model with smaller generalization error.

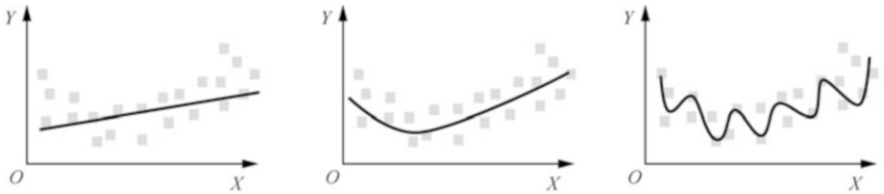


Fig. 2.20 Underfitting, good fitting and overfitting

Once the model is formed and fixed, all possible functions will construct a space, which is called hypothesis space. The machine learning algorithm can be seen as an algorithm that searches for a suitable fitting function in the hypothesis space. A mathematical model that is too simple, or the training time is too short, will cause an increasing training error for the model. This phenomenon is called underfitting. For the former, it should use a more complex model for retraining; for the latter, it only needs to extend the time to effectively eliminate underfitting. However, to accurately determine the cause of under-fitting often requires certain experience and methods. On the contrary, if the model is too complex, it may lead to a small training error, but a weaker generalization ability, which means a larger generalization error known as overfitting. There are many methods to reduce over-fitting. The common ones include appropriately simplifying the model, ending training before the over-fitting occurs, and using dropout and weight decay. Figure 2.20 shows the results of underfitting, good fitting and overfitting for the same dataset.

The capacity of a model refers to its ability to fit a variety of functions, also known as the complexity of a model. When the capacity is compatible for the complexity of the task and the amount of training data provided, the algorithm will usually have the best performance. A model with insufficient capacity cannot handle the complex tasks, thus underfitting may be provoked. As shown in Fig. 2.20a, the data distribution is in a shape of hook, but the model is linear and cannot describe the data distribution properly. A model with a high capacity can handle complex tasks, but when the capacity surpasses the level that the task needs, overfitting may be provoked. As shown in Fig. 2.20c, the model tries to fit the data with a very complex function. Although the training error is reduced, it can be inferred that such a model cannot predict the target value of a new sample properly. The effective capacity of the model is limited by algorithms, parameters, and regularization methods.

Generally speaking, the generalization error can be interpreted as:

Total error = *Bias 2* + *Variance* + *Unresolvable error*

Among them, bias and variance are two sub-forms that we need to pay attention to. As shown in Fig. 2.21, the Variance is the degree of deviation of the model's prediction results near the mean, which is an error derived from the sensitivity of the model to small fluctuations on the training set. Bias is the difference between the average value of the model's prediction results and the correct value we are trying to predict. The unresolvable error refers to the error caused by the imperfection of the

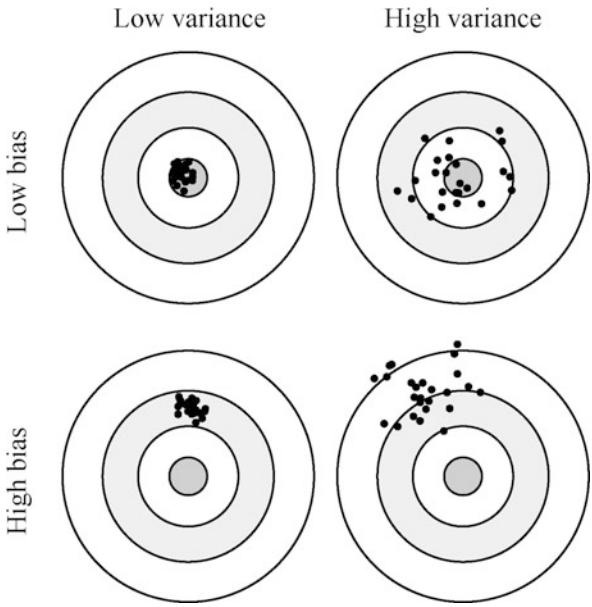


Fig. 2.21 Variance and bias

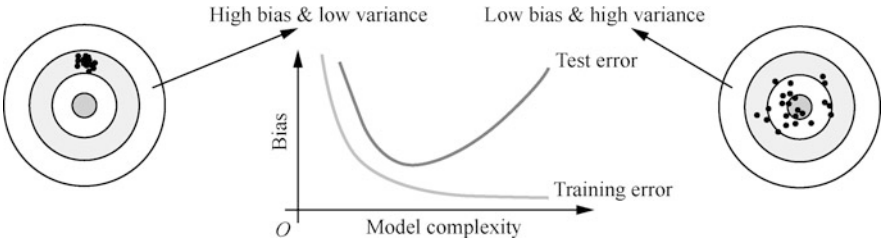


Fig. 2.22 The relationship between model complexity and error

model and the finiteness of the data. In theory, if there is an infinite amount of data and a perfect model, the so-called unresolvable errors can be resolved. But in fact, it is impossible to realize, so the generalization error can never be eliminated.

Ideally, we want to choose a model that can accurately capture the laws in the training data and can also summarize the invisible data (the so-called new data). However, generally speaking, it is impossible for us to accomplish these two things at the same time. As shown in Fig. 2.22, as the complexity of the model increases, the training error gradually decreases. At the same time, the test error will decrease to a certain point as the complexity increases, and then increase in the opposite direction, forming a concave curve. The lowest point of the test error curve suggests the ideal level of model complexity.

When measuring the performance of the regression model, commonly used indicators include mean absolute error (MAE), mean square error (MSE), and correlation coefficient R^2. Assuming that the true target values of the test example

Fig. 2.23 Confusion matrix for a binary classifier

Actual \ Predicted	yes	no	In total
yes	*TP*	*FN*	*P*
no	*FP*	*TN*	*N*
In total	*P′*	*N′*	*P*+*N*

are y1, y2,. . ., ym, and the corresponding predicted values are $\widehat{y}_1, \widehat{y}_2, \cdots, \widehat{y}_m$, then the definition of the above indicators is as follows:

$$MAE = \frac{1}{m}\sum_{i=1}^{m}|y_i - \widehat{y}_i|$$

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y_i - \widehat{y}_i)^2$$

$$R^2 = 1 - \frac{RSS}{TSS} = 1 - \frac{\sum_{i=1}^{m}(y_i - \widehat{y}_i)^2}{\sum_{i=1}^{m}(y_i - \overline{y}_i)^2}$$

Where, TSS represents the difference between the sample values, and RSS represents the difference between the predicted value and the sample value. The values of the MAE and MSE indicators are both non-negative, and the closer to 0, the better the performance of the model. The value of R2 is not greater than 1, and the closer to 1, the better the performance of the model.

When evaluating the performance of a classification model, a method called confusion matrix is often used, as shown in Fig. 2.23. The confusion matrix is a k-dimensional square matrix, where *k* represents the number of all categories. The value in the *i*-th row and the *j*-th column in Fig. 2.23 represents the number of samples that are actually the *i*-th type but are judged to be the *j*-th type by the model. Ideally, for a classifier with higher accuracy, most of the examples should be represented by the diagonal of the confusion matrix, while other values are 0 or close to 0. For the two-classifier confusion matrix shown in Fig. 2.23, the definition of each symbol is as follows.

1. Positive tuple P: tuple of the major classes of interest.
2. Negative tuple N: other tuples except P.
3. True positive example TP: positive tuples correctly classified by the classifier.
4. True negative example TN: the negative tuple correctly classified by the classifier.

Measurement	Equation
Accuracy rate, recognition rate	$\frac{TP+TN}{P+N}$
Error rate, misclassification rate	$\frac{FP+FN}{P+N}$
True positive rate and recall rate	$\frac{TP}{P}$
True negative rate	$\frac{TN}{P}$
Precision rate	$\frac{TP}{TP+FP}$
F_1 value (harmonic mean of precision and recall)	$\frac{2\times precision\times recall}{precision+recall}$
F_β value (β is non-negative real number)	$\frac{(1+\beta^2)\times precision\times recall}{\beta^2\times precision+recall}$

Fig. 2.24 The other concepts in confusion matrix for a binary classifier

5. False positive example FP: a negative tuple that is incorrectly marked as a positive tuple.
6. False negative example FN: A positive tuple that is incorrectly marked as a negative tuple.

Figure 2.24 shows the rest concepts in the binary classifier confusion matrix.

Here let us cite the example of document retrieval to clarify the concepts of precision and recall. The precision describes the proportion of documents that are truly related to the subject among all the documents retrieved. The recall describes the retrieved documents related to the search subject, and the proportion of all related documents in the library.

At the end of this section, let's take an example to illustrate the calculation of the confusion matrix of the binary classifiers. Assuming that a classifier can identify whether there is a cat in an image and 200 images are now used to verify the performance of this model. Among them, 170 are labeled as images with cats and 30 are labeled not. The performance of the model is as shown in Fig. 2.25. It can be seen that the recognition result of the model is that 160 images are marked with cats and 40 pictures not. It can be calculated that the precision of the model is 140/160 = 87.5%, the recall is 140/170 = 82.4%, and the accuracy is (140 + 10)/200 = 75%.

Actual \ Predicted	yes	no	In total
yes	140	30	170
no	20	10	30
In total	160	40	200

Fig. 2.25 Cases of confusion matrix

2.4 Model Parameters and Hyperparameters

Parameters, as part of what the model has learned from historical training data, are the key to machine learning algorithms. Generally speaking, the model parameters are not manually set by the researchers, but are obtained by data estimation or data learning. Identifying the parameter values of the model is equivalent to defining the function of the model, so the model parameters are usually saved as part of the learning model. When implementing model predictions, parameters are also an indispensable component. Examples of model parameters include weights in artificial neural networks, support vectors in support vector machines, and coefficients in linear regression or logistic regression.

There are not only parameters but also hyperparameters in the model. Different from parameters, hyperparameters are external configurations of the model and are often used in the process of estimating model parameters. The most fundamental difference between the two is that the parameters are automatically learned by the model, while the hyperparameters are manually engineered. When handling different prediction modeling problems, it is usually necessary to adjust the model hyperparameters. In addition to being directly specified by the researcher, model hyperparameters can also be set using heuristic methods. Common model hyperparameters include the penalty coefficient in Lasso or ridge regression, the learning rate, number of iterations, batch size, activation function, and number of neurons in the training neural network, the C and σ of the support vector machine, and the K in *KNN*, the number of decision tree models in the random forest, etc.

Model training generally refers to optimizing model parameters, and this process is completed by a gradient descent algorithm. According to the training effect of the model, a series of hyperparameter search algorithms can be used to optimize the hyperparameters of the model. This section first introduces the gradient descent algorithm, and then the concept of the validation set, and then elaborates on the hyperparameter search algorithm and cross-validation.

2.4.1 Gradient Descent

The optimization idea of the gradient descent algorithm is to use the negative gradient direction of the current position as the search direction, which is the fastest descent direction of the current position, as shown in Fig. 2.26a. The formula for gradient descent is as follows:

$$w_{k+1} = w_k - \eta \nabla f_{wk}(x)$$

Where, η is called the learning rate, and w represents the parameters of the model. As w gets closer to the target value, the amount of change in w gradually decreases. When the value of the objective function barely changes or reaches the maximum number of iterations of gradient descent, then it reaches algorithm convergence. It is worth noting that when using the gradient descent algorithm to find the minimum value of a non-convex function, different initial values may lead to different results, as shown in Fig. 2.26b.

When applying gradient descent to model training, multiple variants can be used. Batch Gradient Descent (BGD) uses the gradient mean of the samples in all datasets at the current point to update the weight parameters. Stochastic Gradient Descent (SGD) randomly selects a sample in a dataset, and updates the weight parameters through the gradient of this sample. Mini-batch Gradient Descent (MBGD) combines the characteristics of BGD and SGD, and selects the gradient mean of n samples in the dataset to update the weight parameters each time. Figure 2.27 shows the different performances of the three variants of gradient descent. Among them, the bottom-up curve corresponds to BGD, the top-down curve corresponds to SGD, and the right-to-left curve corresponds to MBGD. BGD is the most stable at runtime, but because every update needs to traverse all samples, it consumes a lot of computing resources. Each update of SGD randomly selects samples, although it improves the computational efficiency, it also brings instability, which may cause the loss function to produce turbulence or even reverse displacement during the

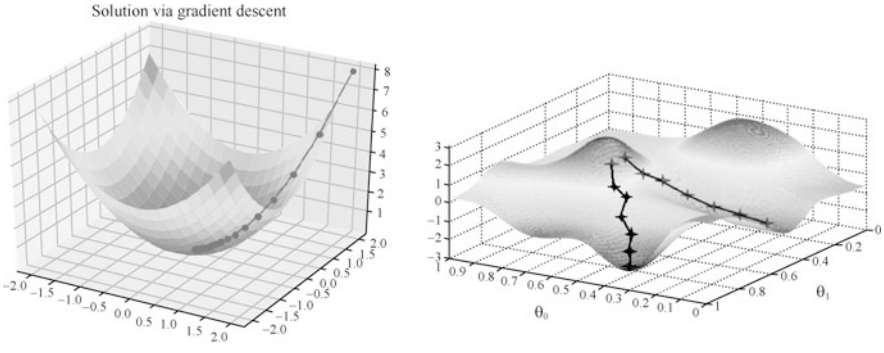


Fig. 2.26 Gradient descent algorithm

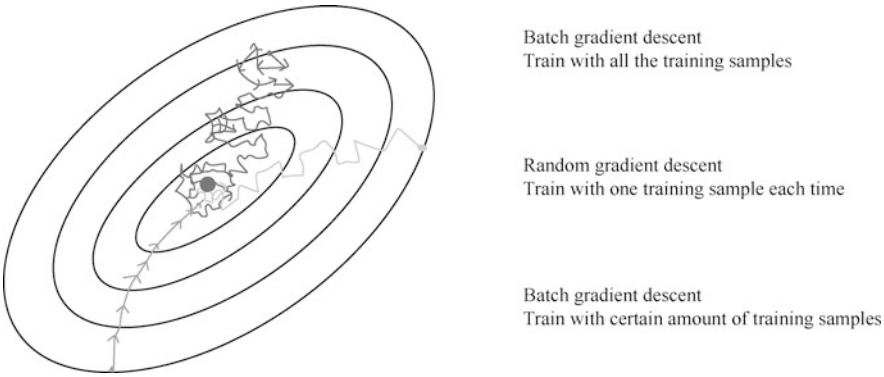


Fig. 2.27 Comparison of the efficiency of gradient descent algorithms

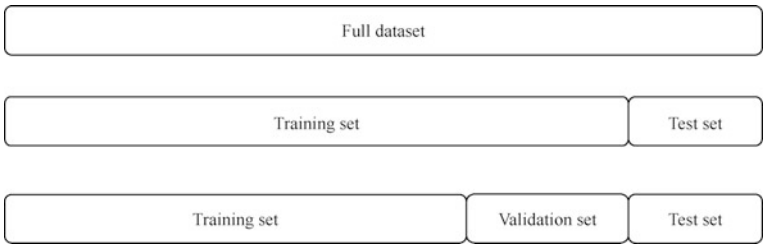


Fig. 2.28 Training set, validation set and test set

process of dropping to the lowest point. MBGD is a method after SGD and BGD are balanced, and it is also the most commonly used gradient descent algorithm in machine learning.

2.4.2 Validation Set and Hyperparameter Search

The training set is a collection of samples used in model training. During the training process, the gradient descent algorithm will try to improve the model's prediction accuracy for the samples in the training set. This causes the model to perform better on the training set than on the unknown dataset. In order to measure the generalization ability of the model, people often randomly select a part of the entire dataset as a test set before training, as shown in Fig. 2.28. The samples in the test set are not involved in training, so they are unknown to the model. It can be approximated that the performance of the model on the test set is the performance of the model under unknown samples.

The optimization goal of hyperparameters is to improve the generalization ability of the model. The most intuitive idea is to try different hyperparameter values, evaluate the performance of these models on the test set, and select the model with

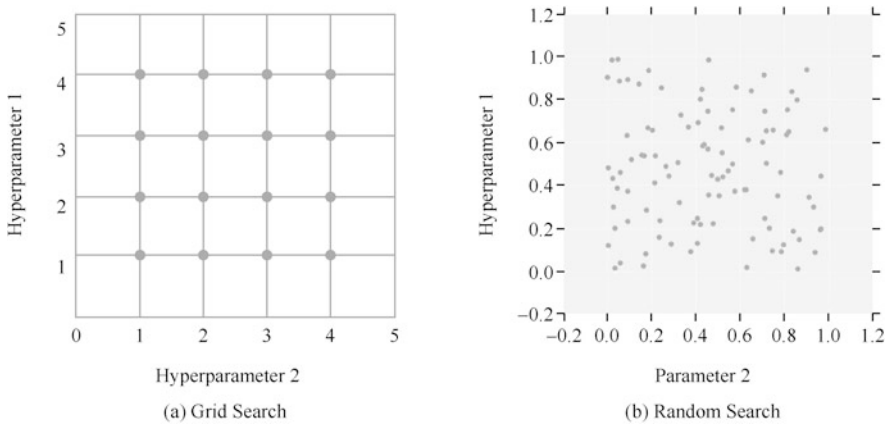


Fig. 2.29 Grid search and random search

the strongest generalization ability. The problem is that the test set cannot participate in model training in any form, even for hyperparameter search. Therefore, some samples should be randomly selected from the training set, and the set of these samples is called the validation set. The samples of the validation set also do not participate in training, and are only used to verify the effect of hyperparameters. Generally speaking, the model needs to be optimized repeatedly on the training set and validation set to finally determine the parameters and hyperparameters and be evaluated on the test set. Methods commonly used to search model hyperparameters include grid search, random search, heuristic intelligent search, and Bayesian search.

Grid search attempts to exhaustively search for all possible hyperparameter combinations to form a grid of hyperparameter values, as shown in Fig. 2.29a. In practice, the range and step length of the grid often need to be manually designated. In the case of a relatively small number of hyperparameters, grid search is applicable, so grid search is feasible in general machine learning algorithms. However, in the case of neural networks, grid search is too expensive and time-consuming, so it is not adopted in most cases.

In the case of a large hyperparameter search space, the result of using random search will be better than grid search, as shown in Fig. 2.29b. Random search implements random sampling of parameters, where each setting is to sample from the distribution of possible parameter values, trying to find the best subset of parameters. To use random search, you need to "coarse adjustment" first and then "fine adjustment". Namely, searching in a coarse range first, and then narrowing the search range according to the position where the best result appears. It is worth noticing that some hyperparameters may be more important than others in actual operation. In this case, the most important hyperparameters will directly affect the search bias, while the secondary hyperparameters may not be well optimized.

2.4.3 Cross-validation

The above-mentioned method of dividing the verification set has two main problems: the chance of sample division is great, and the verification result is not convincing; and the number of samples that can be used for model training is further reduced. In order to solve this problem, the training set can be divided into k groups for *k*-fold cross-validation. K-fold cross-validation will perform *k* rounds of training and verification, where one set of data is used as the verification set in turn, and the remaining $k - 1$ sets of data are used as the training set. This will get *k* models and their classification accuracy on the validation set. The average of these *k* classification accuracy rates can be used as a performance indicator for the generalization ability of the model.

K-fold cross-validation can avoid contingency in the process of dividing the validation set, and the validation results are more convincing. However, using *k*-fold cross-validation requires training *k* models. If the dataset is large, the training will be time-consuming. Therefore, *k*-fold cross-validation is generally applicable to smaller datasets.

The *k* value in *k*-fold cross-validation is also a hyperparameter, which needs to be determined through experiments. In an extreme case, the value of *k* is the same as the number of samples in the training set. This approach is called leave-one-out cross-validation, because one training sample is left as a validation set during each training. The training result of leaving-one-out cross-validation is better, because almost all training samples are involved in the training. But leaving-one-out cross-validation takes a longer time, so it is only suitable for small dataset.

2.5 Common Algorithms of Machine Learning

As shown in Fig. 2.30, there are many common algorithms for machine learning, and a detailed introduction of these algorithms may take a long as a whole book. Therefore, this section only briefly introduces the principles and basic ideas of these algorithms. Readers who are interested in this topic can refer to other books for in-depth understanding.

2.5.1 Linear Regression

Linear regression is a statistical analysis method that uses regression analysis in mathematical statistics to determine the quantitative relationship between two or more variables. It belongs to supervised learning. As shown in Fig. 2.31, the model function of linear regression is a hyperplane:

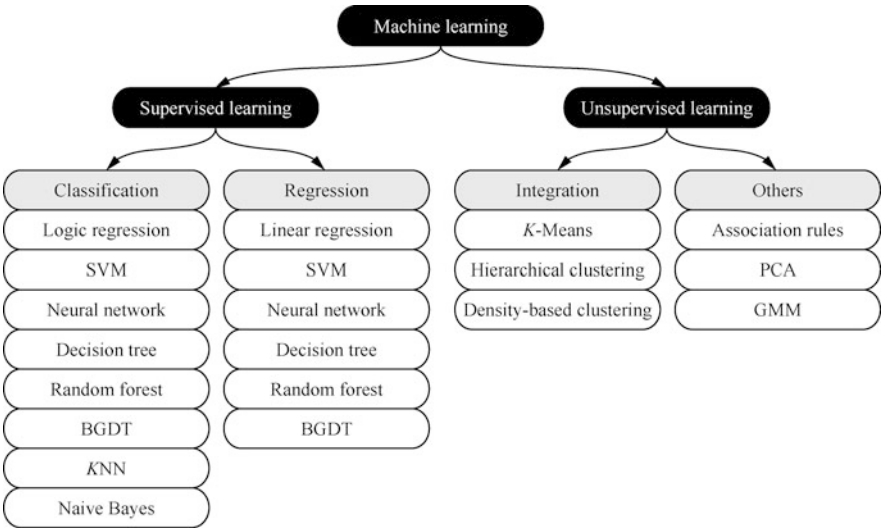
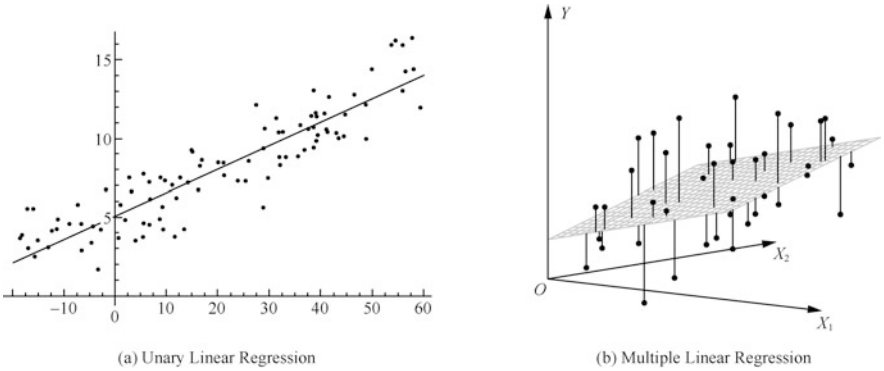


Fig. 2.30 Common algorithms of machine learning



(a) Unary Linear Regression (b) Multiple Linear Regression

Fig. 2.31 Linear regression

$$h(x) = w^{\mathrm{T}}x + b$$

Where, w is the weight parameter, b is the bias, and x is the sample.

The relationship between the predicted value of the model and the true value is as follows:

$$y = h(x) + \varepsilon$$

Where, y represents the true value and represents the error. The error is affected by many factors. According to the central limit theorem, the error obeys the normal distribution.

$$\varepsilon \sim N\left(0, \sigma^2\right)$$

Where, the probability distribution of the true value can be obtained.

$$y \sim N\left(h(x), \sigma^2\right)$$

According to the maximum likelihood estimation, the goal of model optimization is

$$\arg\max_h \prod_{i=1}^{m} P(Y = y_i | X = x_i) = \arg\max_h \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(h(x_i) - y_i)^2}{2\sigma^2}\right)$$

Where, argmax represents the maximum point, which is h that maximizes the value of the objective function. In the objective function, $\left(\sqrt{2\pi}\sigma\right)^{-1}$ is a constant that has nothing to do with h, and multiplying or dividing the objective function by a constant will not change the position of the maximum point, so the optimization objective of the model can be transformed into

$$\arg\max_h \prod_{i=1}^{m} \exp\left(-\frac{(h(x_i) - y_i)^2}{2\sigma^2}\right)$$

Because the logarithmic function is monotonic, taking ln for the objective function will not affect the maximum point.

$$\arg\max_h \ln\left(\prod_{i=1}^{m} \exp\left(-\frac{(h(x_i) - y_i)^2}{2\sigma^2}\right)\right) = \arg\max_h \sum_{i=1}^{m} -\frac{(h(x_i) - y_i)^2}{2\sigma^2}$$

By taking the negative of the objective function, the original maximum point will become the minimum point. At the same time, we can also multiply the objective function by a constant to convert the optimization goal of the model into

$$\arg\min_h \frac{1}{2m} \sum_{i=1}^{m} (h(x_i) - y_i)^2$$

Obviously, the loss function is

$$J(w) = \frac{1}{2m} \sum_{i=1}^{m} (h(x_i) - y_i)^2$$

We hope that the predicted value is as close as possible to the true value, that is, to minimize the loss value. The method of gradient descent can be used to find the

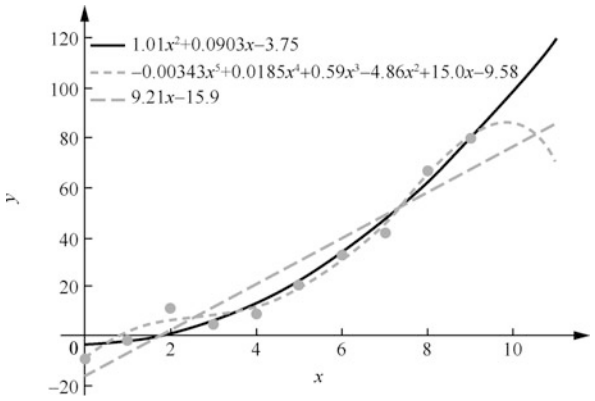


Fig. 2.32 Comparison of linear regression and polynomial regression

weight parameter w when the loss function is minimized, and then the model construction is completed.

Polynomial regression is a branch of linear regression. Generally, the complexity of the dataset would exceed the possibility of fitting with a straight line, that is, using the original linear regression model will obviously underfit. The solution is to use polynomial regression, as shown in Fig. 2.32, the formula is

$$h(x) = w_1 x + w_2 x^2 + \cdots + w_n x^n + b$$

Where, n represents the polynomial regression dimension.

The polynomial regression dimension is a hyperparameter. If you choose it carelessly, it may cause overfitting. Applying regularization helps reduce overfitting. The most common regularization method is to add a square sum loss on top of the objective function

$$h(x) = w_1 x + w_2 x^2 + \cdots + w_n x^n + b$$

Where $\|\bullet\|_2$ represents the L2 regular term. The linear regression model using this loss function is also called the ridge regression model. Similarly, the linear regression model with the added absolute value loss is called the Lasso regression model, and its formula is

$$J(\boldsymbol{w}) = \frac{1}{2m} \sum_{i=1}^{m} (h(x_i) - y_i)^2 + \lambda \sum \|\boldsymbol{w}\|_1$$

Where $\|\bullet\|_1$ represents the L1 regular term.

2.5.2 Logistic Regression

Logistic regression model is a classification model used to solve classification problems. The definition of the model is as follows:

$$h(x) = P(Y = 1|X) = g\left(\boldsymbol{w}^{\mathrm{T}}x + b\right)$$

Where g represents the sigmoid function, w represents the weight, and b, the bias. In the formula, is a linear function of x, so logistic regression, like linear regression, belongs to the generalized linear model.

The definition of the sigmoid function is as follows:

$$g(x) = \frac{1}{1 + \exp\{-x\}}$$

The image of the sigmoid function is shown in Fig. 2.33.

By comparing the magnitude relationship between P(Y = 1|X) and the threshold t, the classification result corresponding to x can be obtained. The threshold t here is a hyperparameter of the model, which can be chosen arbitrarily. It can be seen that when the threshold is large, the model tends to judge the sample as a negative example, so the precision rate will be higher; when the threshold is smaller, the model tends to judge the sample as a positive example, so the recall rate will be higher. Generally, 0.5 can be used as the threshold.

According to the idea of maximum likelihood estimation, when the sample is a positive example, we expect P(Y = 1|X) to be larger; when the sample is a negative example, we expect P(Y = 0|X) to be larger. In other words, we expect the following equation to be as large as possible whatever the sample is:

$$P = P(Y = 1|X)^{y} P(Y = 0|X)^{1-y}$$

Replace P(Y = 1|X) and P(Y = 0|X) with h(x) to get

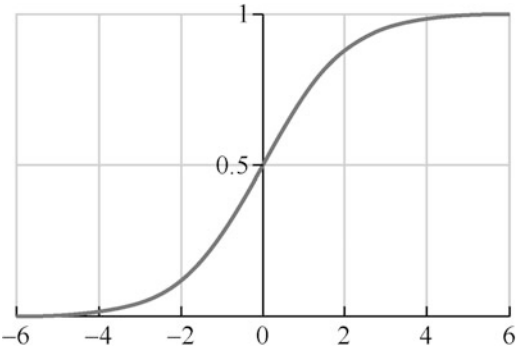


Fig. 2.33 Sigmoid function

$$P = h(x)^y \cdot (1 - h(x))^{1-y}$$

Therefore, the goal of model optimization is

$$\arg\max_h \prod_{i=1}^{m} P_i = \arg\max_h \prod_{i=1}^{m} h(x)^y (1 - h(x))^{1-y}$$

The derivation process similar to linear regression can take the logarithm of the objective function without changing the position of the maximum point. Therefore, the optimization goal of the model is equivalent to

$$\arg\max_h \sum_{i=1}^{m} (y \ln h(x) + (1 - y) \ln (1 - h(x)))$$

Multiplying the objective function by the constant −1/m will cause the original maximum point to become the minimum value point, which is

$$\arg\min_h \frac{-1}{m} \sum_{i=1}^{m} (y \ln h(x) + (1 - y) \ln (1 - h(x)))$$

Therefore, the loss function of logistic regression is

$$J(\boldsymbol{w}) = -\frac{1}{m} \sum (y \ln h(x) + (1 - y) \ln (1 - h(x)))$$

Where, w represents the weight parameter, m is the number of samples, x is the sample, and y is the true value. The value of the weight parameter w can also be obtained through the gradient descent algorithm.

Softmax regression is a generalization of logistic regression, which is applicable for k classification problems. Essentially, the softmax function compresses (maps) a k-dimensional arbitrary real number vector into another k-dimensional real number vector to represent the probability distribution of the category of the sample. The softmax regression probability density function is as follows:

$$P(Y = c|x) = \frac{\exp\left\{w_c^{\mathrm{T}} x + b\right\}}{\sum_{l=1}^{k} \exp\left\{w_c^{\mathrm{T}} x + b\right\}}$$

As shown in Fig. 2.34, the softmax function assigns probability values to each category in the multi-class problem, and these probabilities add up to 1. Among these categories, the probability value of the sample category being apple is the largest, which is 0.68, so the predicted value of the sample should be the apple.

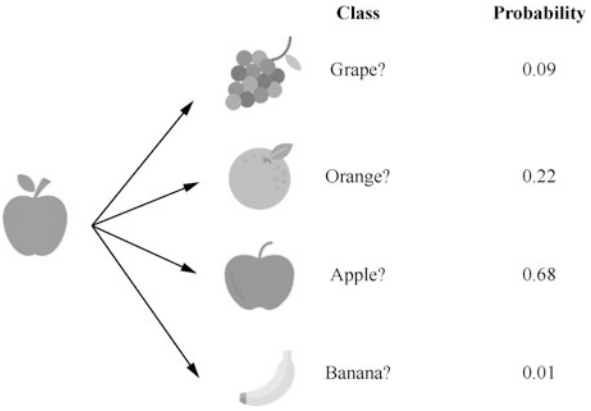


Fig. 2.34 An example of the softmax function

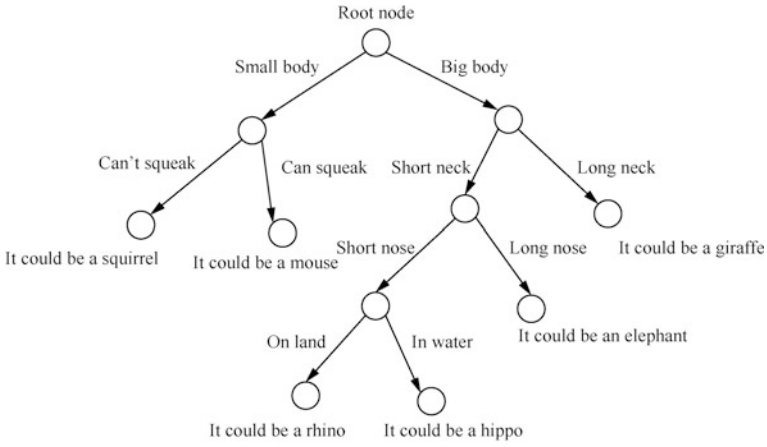


Fig. 2.35 An example of decision tree

2.5.3 Decision Tree

Decision tree is a tree structure (binary or non-binary) classifier, as shown in Fig. 2.35. Each non-leaf node represents a test on a feature attribute, each branch represents the output of this feature attribute in a certain value range, and each leaf node stores a category. The process of using a decision tree to make a decision is to start from the root node, test the corresponding feature attributes in the items to be classified, and select the output branch according to its value until the leaf node is reached, and the category stored in the leaf node is used as the decision result.

The most important thing in the decision tree model is the structure of the tree. The construction of the so-called decision tree is to select attributes to determine the topological structure between each feature attribute. The key step in constructing a decision tree is to perform the division operation according to all the feature

NO.	Tax refund	Marital Status	Taxable income/yuan	Tax fraud
1	Yes	Unmarried	125k	No
2	No	Married	100k	No
3	No	Unmarried	70k	No
4	Yes	Married	120k	No
5	No	Divorced	95k	Yes
6	No	Married	60k	No
7	Yes	Divorced	220k	No
8	No	Unmarried	85k	Yes
9	No	Married	75k	No
10	No	Unmarried	90k	Yes

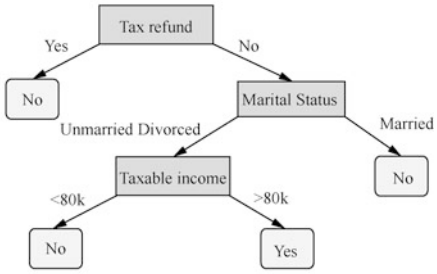


Fig. 2.36 Building a decision tree

attributes, compare the purity of the result set of all the division operations, and select the attribute with the highest purity as the data point of the split dataset. The learning algorithm of the decision tree is the algorithm that constructs the decision tree, and the commonly used algorithms include ID3, C4.5 and CART. The difference between these algorithms is mainly in the quantitative indicators of purity, such as information entropy and Gini coefficient:

$$H(X) = -\sum_{k=1}^{K} p_k \log_2 p_k$$

$$\text{Gini} = 1 - \sum_{k=1}^{K} p^2{}_k$$

Where, pk represents the probability that the sample belongs to class k, and K represents the total number of categories. The greater the difference in purity before and after segmentation, the more conducive that judging a certain feature is to the improvement of the accuracy of the model, indicating that it should be added to the decision tree model.

In general, the process of building a decision tree consists of the following three stages.

1. Feature selection: select a feature from the features of the training data as the split criterion for the current node (different criteria for feature selection produce different decision tree algorithms).
2. Decision tree generation: According to the selected feature evaluation criteria, child nodes are generated recursively from top to bottom until the dataset is inseparable, then the decision tree growth is stopped.
3. Pruning: By reducing the size of the tree to suppress the overfitting of the model, it can be divided into pre-pruning and post-pruning.

Figure 2.36 shows a case of classification using a decision tree model. The classification result is affected by three attributes: tax refund, marital status and taxable

income. From this example, we can see that the decision tree model can not only handle the case where the attribute takes two values, but also the case where the attribute takes multiple values or even continuous values. In addition, the decision tree model is interpretable, and we can intuitively analyze the importance relationship between attributes based on the structure chart shown in Fig. 2.36b.

2.5.4 Support Vector Machine

Support vector machine (SVM) is a linear classifier with the largest interval defined in the feature space. The learning algorithm of SVM is an optimal algorithm for solving convex quadratic linear programming. In summary, the core concepts of SVM include the following two aspects.

1. Search for the optimal hyperplane in the feature space based on the structural risk minimization theory, so that the learner obtains global optimization, and the expectation in the entire sample space satisfies a certain upper bound with a certain probability.
2. For linearly inseparable data, map the linearly inseparable samples of the low-dimensional input space to the high-dimensional feature space to make them linearly separable based on a nonlinear mapping algorithm, so that the high-dimensional feature space adopts the linear algorithm for the nonlinearity of the sample Linear analysis of features becomes possible.

Straight lines are used to divide the data into different categories, but in fact we can find multiple straight lines to separate the data, as shown in Fig. 2.37. The core idea of SVM is to find a straight line that meets the above conditions, and make the points closest to the straight line as distant as possible from this straight line. This will give the model a strong generalization ability. These points closest to the straight line are called support vectors.

Linear SVM can perform properly on linear separable datasets, but we cannot use straight lines to divide non-linear datasets. At this time, a kernel function is needed to construct a nonlinear SVM. The kernel function allows the algorithm to fit the hyperplane in the transformed high-dimensional feature space, as shown in

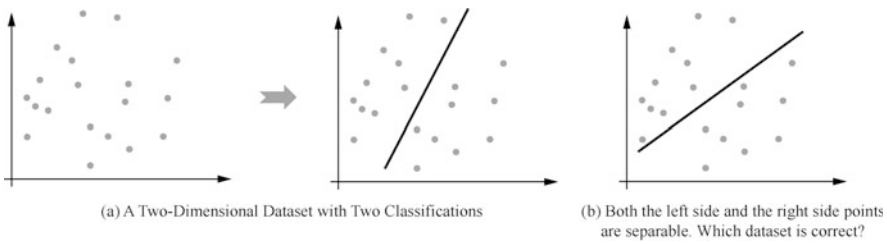


Fig. 2.37 Performance of linear classifier

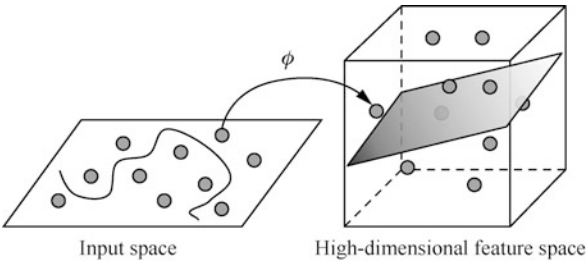


Fig. 2.38 Kernel function

Fig. 2.38. Common kernel functions include linear kernel function, polynomial kernel function, Sigmoid kernel function and Gaussian kernel function. The Gaussian kernel function can map samples to infinite dimensional space, so the effect is also better, and it is one of the most commonly used kernel functions.

2.5.5 K-*Nearest Neighbor Algorithm*

The *K*-nearest neighbor (KNN) algorithm is a theoretically mature method and one of the simplest machine learning algorithms. The *K*NN algorithm is a non-parametric method, which tends to perform better in datasets with irregular decision boundaries. The idea of this method is: if most of the *K* nearest samples (i.e., nearest neighbors in the feature space) of a sample in the feature space belong to a certain category, then the sample also belongs to this category.

The core concept of the *K*NN algorithm is "What's around cinnabar goes red, and what's around ink turns black", featuring a concise logic. But like *k*-fold cross-validation, *K* in the *K*NN algorithm is also a hyperparameter. This means that it is difficult to select the *K* value appropriately. As shown in Fig. 2.39, when the *K* value is 3, the prediction result at the question mark will be a triangle; and when the *K* value is 5, the prediction result at the question mark will become a square. Figure 2.40 shows the decision boundary for different *K* values. It can be found that as the value of *K* increases, the decision boundary will become smoother. Generally speaking, a larger *K* value will reduce the impact of noise on classification but will make the boundaries of classes less obvious. The larger the *K* value is, the more likely it is to cause under-fitting, because that the decision boundaries are too blur. Correspondingly, the smaller the *K* value is, the easier it is to cause over-fitting, because that the decision boundaries are too sharp.

The KNN algorithm can not only be used for the classification problems, but also for the regression problems. In the classification prediction problem, it normally adopts the majority voting method. In the regression prediction problem, the average method is widely used. Although these methods are seemingly only about the *K* samples of the nearest neighbors, the computation volume of the *K*NN algorithm

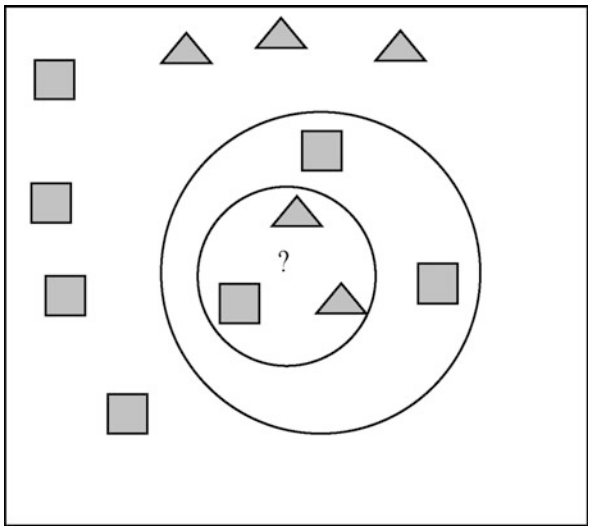


Fig. 2.39 An example of *K*NN algorithm

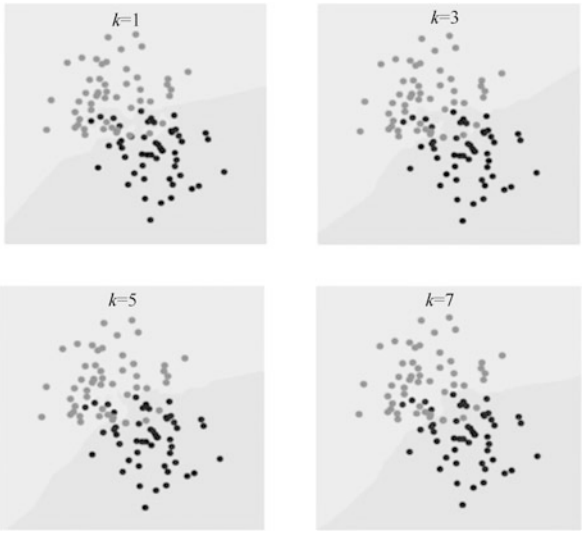


Fig. 2.40 The influence of *K* value on decision boundary

is very large in fact. This is because that the *K*NN algorithm needs to traverse all samples to determine which *K* samples are the nearest neighbors to the sample to be tested.

2.5.6 Naive Bayes

Naive Bayes is a simple multi-classification algorithm based on Bayes' theorem and assumes that the features are independent. Given the sample feature X, the probability that the sample belongs to class c is

$$P(C = c|X) = \frac{P(X|C = c)P(C = c)}{P(X)}$$

Where, $P(C = c|X)$ is called the posterior probability, $P(C = c)$ represents the prior probability of the target, and $P(X)$ represents the prior probability of the feature. Normally we do not consider $P(X)$, because $P(X)$ can be seen as a fixed value when classifying, that is

$$P(C = c|X) \propto P(X|C = c)P(C = c)$$

$P(C = c)$ has nothing to do with X and needs to be determined before training the model. Generally, the proportion of samples with category c in the dataset is calculated as $P(C = c)$. It can be seen that the core of classification is to find $P(X|C = c)$. Suppose feature X is composed of the following elements:

$$X = (X_1, X_2, \cdots, X_n)$$

Generally, it can be easily calculated that

$$\prod_{i=1}^{n} P(X_i|C = c)$$

Combining the attribute conditional independence assumption, we can prove

$$P(X|C = c) = \prod_{i=1}^{n} P(X_i|C = c)$$

The attribute conditional independence assumption states that given the sample classification as a condition, the distribution of each attribute value is independent of the distribution of other attribute values. The reason why Naive Bayes is naive is precisely because of the attribute independence assumption used in its model. Making this assumption effectively simplifies the calculation and gives the Bayesian classifier a higher accuracy and training speed on large databases.

Here is an example. We want to judge a person's gender C by his height X1 and weight X2. Suppose that the probability of a person with a height of 180 cm and a height of 150 cm is male is 80% and 20%, respectively, and the probability of a person with a weight of 80 kg and 50 kg is male is 70% and 30%, respectively.

According to the Naive Bayes model, the probability that a person with a height of 180 cm and a weight of 50 kg is male is $0.8 \times 0.3 = 0.24$, while the probability that a person with a height of 150 cm and a weight of 80 kg is male is only $0.7 \times 0.2 = 0.14$. It can be considered that the two features of height and weight independently contribute to the probability that this person is male.

The performance of the Naive Bayes model usually depends on the degree to which the feature independence hypothesis is satisfied. As mentioned in the previous example, the two features of height and weight are not completely independent. This correlation will inevitably affect the accuracy of the model, but as long as the correlation is not large, we can continue to use the Naive Bayes model. In practical applications, different features are rarely completely independent.

2.5.7 Ensemble Learning

Integrated learning is a machine learning paradigm. In this paradigm, multiple learners are trained and combined to solve the same problem, as shown in Fig. 2.41. Thanks to the multiple learners involved, the generalization ability of ensemble learning can be much stronger than using a single learner. Let's imagine you randomly ask a complicated question to several thousands of people, and then combine their answers together. In most cases, this integrated answer is even better than an answer provided by an expert. This is the collective intelligence we talk about.

The implementation methods of ensemble learning can be classified into two types—bagging and boosting. Bagging independently builds several basic learners, and then averages their predictions. Typical models of Bagging include random

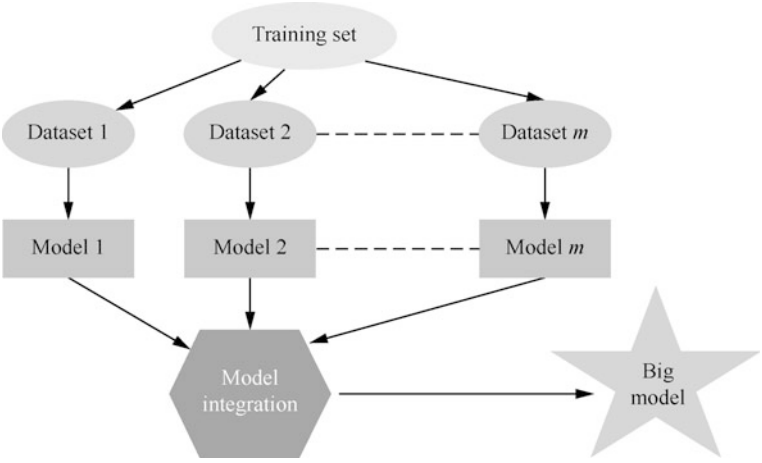


Fig. 2.41 Ensemble learning

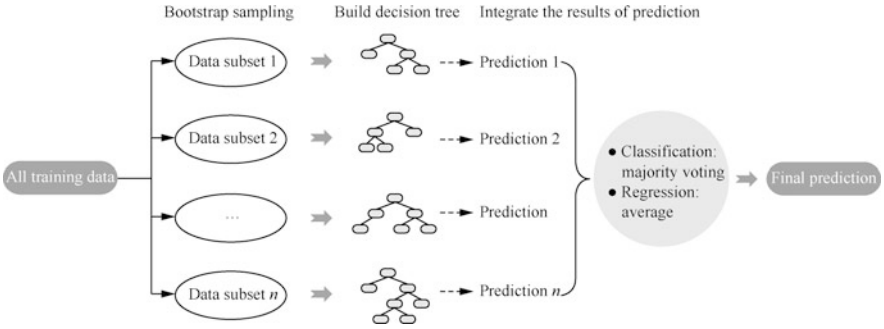


Fig. 2.42 Random forest algorithm

forests and so on. On average, the prediction result of the combined learner is usually better than any single elementary learner because its variance is reduced. Boosting constructs the basic learner in a sequential manner, and gradually reduces the deviation of the comprehensive learner's prediction. Typical models of Boosting include Adaboost, GBDT and XGboost. In general, Bagging can reduce the variance, thereby suppressing over-fitting; while Boosting focuses on reducing the deviation, thereby increasing the capacity of the model, but it may cause over-fitting.

Random forest algorithm is a combination of the bagging method and the CART decision tree. The overall process of the algorithm is shown in Fig. 2.42. Random forest algorithm can be used for classification and regression problems. The basic principle is to build multiple decision trees and merge them to make more accurate and stable predictions. During the training process of the decision tree, sampling is performed at the two levels of sample and feature at the same time. At the sample level, the bootstrap sampling (sampling with replacement) is used to determine the sample subset used for decision tree training. At the feature level, before each node of the decision tree is split, some features are randomly selected to calculate the information gain. By synthesizing the prediction results of multiple decision trees, the random forest model can reduce the variance of a single decision tree model, but the effect of correcting the deviation is not satisfactory. Therefore, the random forest model requires that every decision tree cannot be underfitted, even if this requirement may cause some decision trees to overfit. Also, note that each decision tree model in the random forest is independent, so the training and prediction processes can be performed in parallel.

Gradient boosting decision tree (GBDT) is one of the Boosting methods. The predicted value of the model is the sum of the results of all decision trees. The essence of GBDT is to continuously use new decision trees to learn the residuals of all previous decision trees, that is, the error between the predicted value and the true value. As shown in Fig. 2.43, for a given sample, the prediction result of the first decision tree is 20 years old, while the true age of the sample is 30. The difference between the predicted result and the true value is 10. If we can predict this difference with another decision tree, we can improve the prediction result of 20 and make it closer to 30. Based on this idea, we introduce the second decision tree to learn the

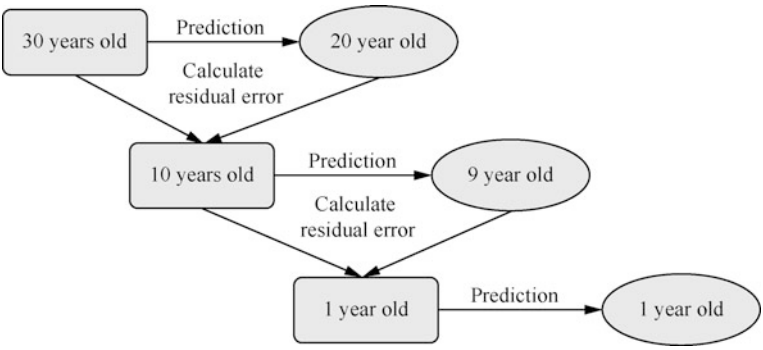


Fig. 2.43 GBDT algorithm

error of the first decision tree, and so on. Finally, the prediction results of the three learners are added together to get the true value of 30. GBDT improves accuracy by continuously correcting the deviation of the decision tree, thus allowing a certain degree of underfitting of the decision tree. However, GBDT cannot correct the variance, so it is generally not allowed to overfit the decision tree. This is also one of the biggest differences between the boosting and bagging methods. In addition, the training data of each decision tree in GBDT depends on the output of the previous decision tree, so the training process cannot be parallelized.

2.5.8 Clustering Algorithm

K-means clustering algorithm (*K*-Means clustering) is an algorithm that inputs the number of clusters *K* and a dataset containing n data objects, and outputs *K* clusters that meet the minimum variance standard, as shown in Fig. 2.44. It shows that the final obtained cluster meets: the similarity of objects in the same cluster is higher; and the similarity of objects in different clusters is lower.

Compared to the *K*-Means algorithm, the hierarchical clustering algorithm also outputs the tree-like relationship between the samples while outputting the clusters. As shown in Fig. 2.45, the hierarchical clustering algorithm tries to divide the dataset at different levels so as to form a tree-shaped clustering structure. The dataset can be divided either by a "bottom-up" agglomerative strategy, or a "top-down" divisive strategy. The hierarchy of clusters is represented as a tree diagram, where the root of the tree represents the ancestor class of all samples, and the leaves are clusters with only one sample.

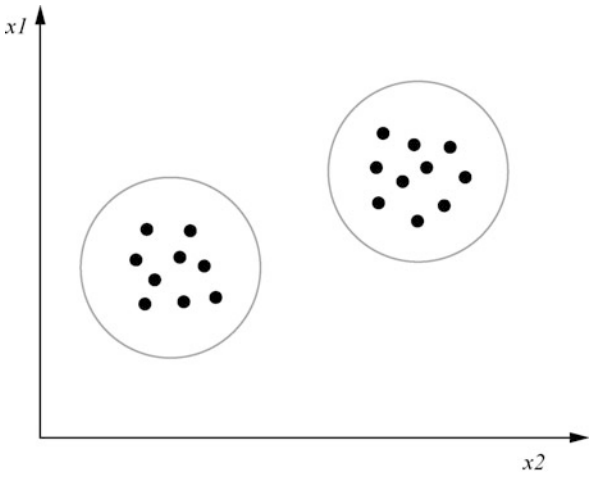


Fig. 2.44 K-Means algorithm

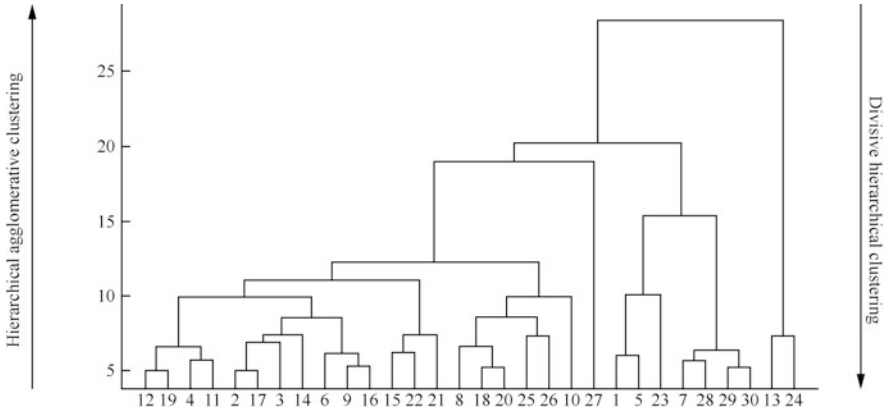


Fig. 2.45 Hierarchical clustering algorithm

2.6 Case Study

By the end of this chapter, we are about to review the overall process of a machine learning project with one case. Suppose there is a dataset that gives the living area (1 square foot ≈ 0.09 square meter) and price of 21,613 houses sold in a certain city, as shown in Fig. 2.46. Based on such data, we hope to train a model to predict the prices of other houses in the city.

It can be inferred from the data in the house price dataset that the input (house area) and output (price) in the data are continuous values, so the regression model in supervised learning can be used. The goal of the project is to build a model function h(x) to make the model infinitely approximate the function that expresses the true

Fig. 2.46 Housing price dataset

x	y
Floor area/square foot	Price/USD
1180	221,900
2570	538,000
770	180,000
1960	604,000
1680	510,000
5420	1,225,000
1715	257,500
1060	291,850
1160	468,000
1430	310,000
1370	400,000
1810	530,000
…	…

Dataset

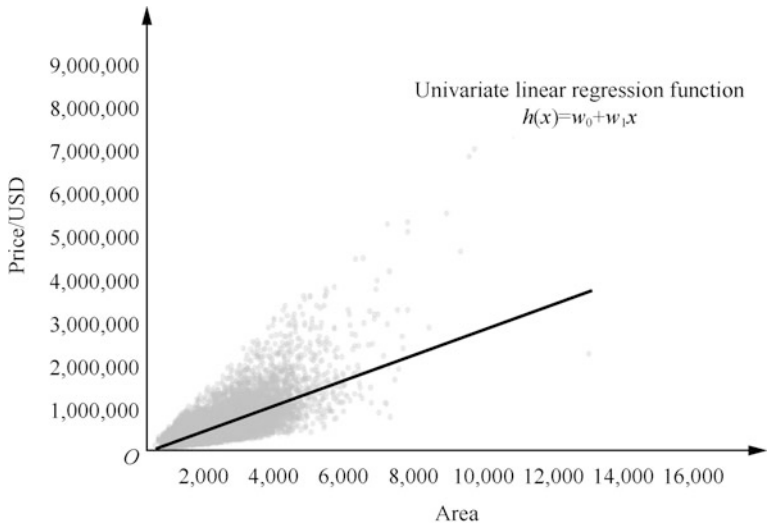


Fig. 2.47 Model assumptions/square foot

distribution of the dataset. Figure 2.47 shows a scatter plot of the data and a possible model function.

The goal of linear regression is to find a straight line that best fits the dataset, that is, to determine the parameters in the model. In order to find the best parameters, we need to construct a loss function and find the parameter value when the loss function reaches the minimum value. The equation of the loss function is as follows:

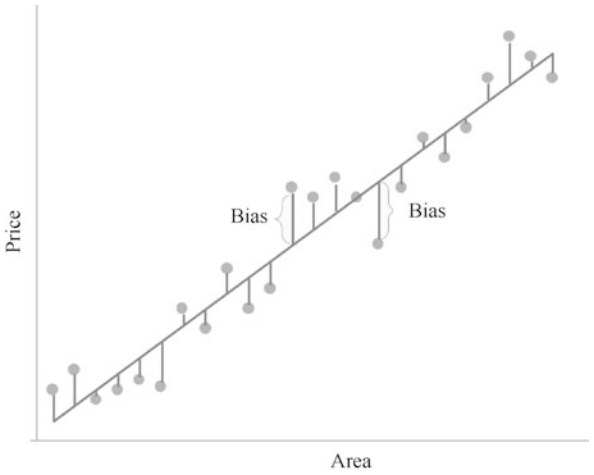


Fig. 2.48 Geometric meaning of error

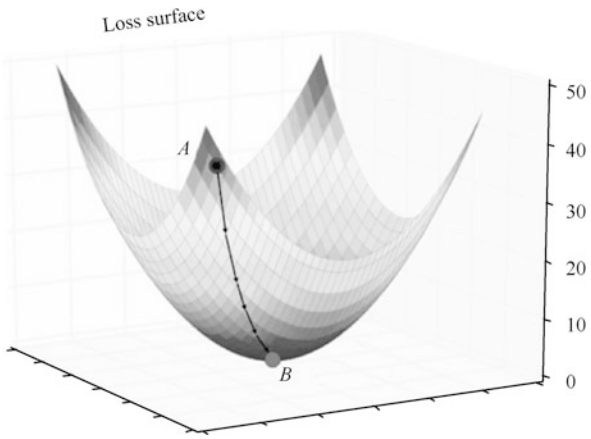


Fig. 2.49 Loss surface

$$J(w) = \frac{1}{2m} \sum (h(x) - y)^2$$

Where m represents the number of samples, h(x) is the predicted value, and y is the true value. Intuitively, the loss function represents the sum of squared errors from all samples to the model function, as shown in Fig. 2.48. When this loss function is reduced to the minimum, all samples should be evenly distributed on both sides of the fitted straight line. At this time, the fitted straight line is the model function we require.

As mentioned earlier, the gradient descent algorithm uses an iterative method to find the minimum value of a function. The gradient descent algorithm first randomly selects an initial point on the loss function, and then finds the global minimum value of the loss function according to the negative gradient direction. The parameter value at this time is the best parameter value we require, as shown in Fig. 2.49. Point

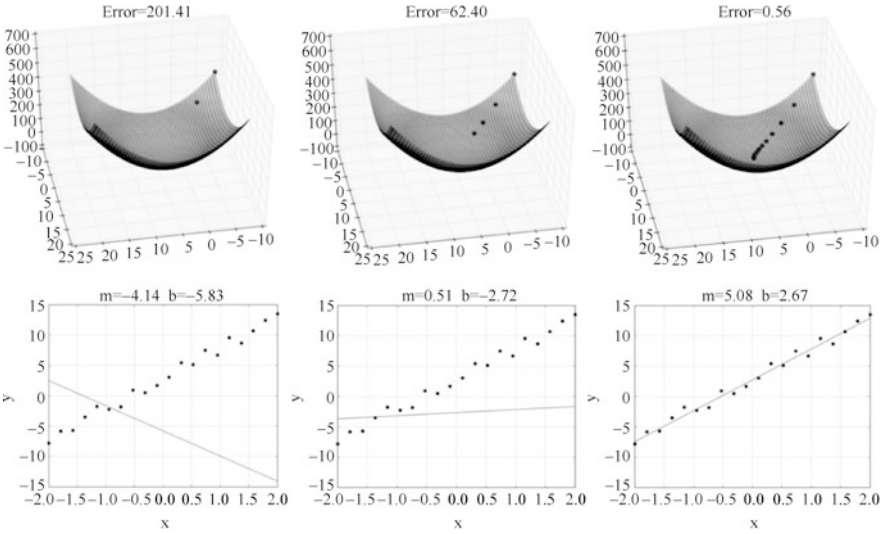


Fig. 2.50 Visualization of the gradient descent process

A represents the position where the parameter *w* is randomly initialized; point *B* represents the global minimum of the loss function, which is the final parameter value; the connected line between *A* and *B* represents the trajectory formed by the negative gradient direction. For every iteration, the value of parameter *w* will change, resulting in the constant change of the regression line.

Figure 2.50 shows an example of an iterative process using gradient descent. It can be observed that as the points on the loss surface gradually approach the lowest point, the linear regression fitting line fits the data better and better. Finally, we can get the best model function $h(x) = 280.62\text{x} - 43{,}581$.

After the model training is completed, we need to use the test set for testing to ensure that the model has sufficient generalization capabilities. If there is overfitting in the test, we can add a regular term to the loss function and adjust the hyperparameters. If it is under-fitting, we can use more complex regression models, such as GBDT. After that, the model needs to be retrained, and the test set is reused for testing until the generalization ability of the model meets expectations. It should be noted that since real data is used in the project, the role of data cleaning and feature selection cannot be ignored either.

2.7 Chapter Summary

This chapter mainly introduces the definition, classification and major challenges of machine learning. Meanwhile, the overall process of machine learning (data collection, data cleaning, feature extraction and selection, model training, model

evaluation and testing, model deployment and integration, etc.), common machine learning algorithms (linear regression, logistic regression, decision trees, support vector machines, naive Bayes, *K*NN, ensemble learning, *K*-Means, etc.), gradient descent algorithms, hyperparameters and other important machine learning knowledge are sorted out and explained; finally, through the use of linear regression, the housing price prediction case is completed, showcasing the overall process of machine learning.

2.8 Exercises

1. Machine learning is the core technology of artificial intelligence. Please tell us the definition of machine learning.
2. The generalization error of the model can be classified into variance, bias, and irresolvable errors. What is the difference between variance and bias? What are the characteristics of the variance and bias of an overfitting model?
3. In accordance with the confusion matrix shown in Fig. 2.25, please find the F1 value.
4. In machine learning, the entire dataset is generally divided into three parts: training set, validation set, and test set. What is the difference between the verification set and the test set? Why introduce a validation set?
5. Linear regression models use linear functions to fit the data. For nonlinear data, how to deal with the linear regression model?
6. Many classification models can only handle two classification problems. Taking SVM as an example, try to find a solution for multi-classification problems.
7. Please refer to the relevant information and answer how does the Gaussian kernel function in SVM map features to infinite dimensional space?
8. Is gradient descent the only way to train the model? What are the limitations of this method?