

Chapter 2

Traditional Machine Learning

Abstract In this chapter, we describe the concepts of traditional machine learning. In particular, we introduce the key features of supervised learning, heuristic learning, discriminative learning, single-task learning and random data partitioning. We also identify general issues of traditional machine learning, and discuss how traditional learning approaches can be impacted due to the presence of big data.

2.1 Supervised Learning

Supervised learning means learning with a teacher, i.e. data used in the training stage must be fully labeled by experts. In practice, supervised learning can be involved in classification and regression tasks. The main difference between classification and regression is that the output attribute must be discrete for the former type of tasks, whereas the output attribute must be continuous for the latter type of tasks. From this point of view, classification is also known as categorical prediction. Similarly, regression is also known as numerical prediction.

From philosophical perspectives, supervised learning is very similar to the case of student learning for which it is necessary to provide students with both questions and answers, such that students can identify if they have grasped the knowledge correctly, prior to exams. In this context, training data is like revision questions and test data is like exam questions.

Although supervised learning leads to straightforward evaluation of performance in a learning task, this type of learning would be constrained greatly in the big data era due to some issues, such as the rapid accumulation of varied data. More details on these issues will be given in Sect. 2.7.

2.2 Heuristic Learning

Heuristic learning means learning with a specific strategy, which is based on statistical heuristics, such as Bayes Theorem (Eq. 2.1) and distance functions (Eqs. 2.2 and 2.3).

$$P(Y|X) = \frac{P(Y)P(X|Y)}{P(X)} \quad (2.1)$$

where X and Y are two events:

- $P(X)$ is read as the probability that event X occurs to be used as evidence supporting event Y .
- $P(Y)$ is read as the prior probability that event Y occurs on its own.
- $P(Y|X)$ is read as the posterior probability that event Y occurs given that event X truly occurs.
- $P(X|Y)$ is read as conditional probability that event X occurs subject to that event Y must occur.

$$D(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (2.2)$$

where n is the dimensionality (the number of attributes) and i is the index of the attribute. For example, if the data is in two dimensions, then Eq. 2.2 can be rewritten as Eq. 2.3 as follows:

$$D(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} \quad (2.3)$$

Bayes' Theorem is essentially used in Bayesian learning, and a popular algorithm of this type of learning is Naive Bayes (NB), as mentioned in Chap. 1. The learning outcome of this algorithm is finding the class that has the highest posterior probability given all input attributes with their values as the joint condition. The learning strategy of this algorithm is in the following procedure based on Bayes' theorem:

Step 1: Calculating the posterior probability of each class given each attribute with its value $P(y = c_k | x_i = v_{ij})$ on the basis of training instances, where y is the class attribute, c_k is the k^{th} class label of y , x is the input attribute, i is the index of attribute x and v_{ij} is the j^{th} value of x_i .

Step 2: Calculating the posterior probability of a class: $\prod_{i=1}^n P(y = c | x_i = v_i)$, where i is the index of attribute x and n is the number of attributes.

Step 3: Assigning the test instance the class that has the highest posterior probability on the basis of all the attribute values.

An example shown in Table 2.1 is used for illustrating the procedure of the NB algorithm:

Table 2.1 Illustrative Example for Naive Bayes [1]

x_1	x_2	x_3	y
0	0	1	0
0	1	0	1
1	0	1	1
1	1	0	0
0	1	1	1

Following Step 1 of the procedure illustrated above, the posterior probability of each class given each attribute with its value is listed below:

$$\begin{aligned}
P(y = 0|x_1 = 0) &= 0.33, P(y = 1|x_1 = 0) = 0.67, \\
P(y = 0|x_1 = 1) &= 0.50, P(y = 1|x_1 = 1) = 0.50, \\
P(y = 0|x_2 = 0) &= 0.50, P(y = 1|x_2 = 0) = 0.50, \\
P(y = 0|x_2 = 1) &= 0.33, P(y = 1|x_2 = 1) = 0.67, \\
P(y = 0|x_3 = 0) &= 0.50, P(y = 1|x_3 = 0) = 0.50, \\
P(y = 0|x_3 = 1) &= 0.33, P(y = 1|x_3 = 1) = 0.67,
\end{aligned}$$

Following Step 2 of the above procedure, the value of y , when $x_1 = 1$, $x_2 = 1$ and $x_3 = 1$, is essentially calculated in the following way:

$$\begin{aligned}
P(y = 0|x_1 = 1, x_2 = 1, x_3 = 1) &= P(y = 0|x_1 = 1) \times P(y = 0|x_2 = 1) \times P(y = 0|x_3 = 1) \\
&= 0.5 \times 0.33 \times 0.33; \\
P(y = 1|x_1 = 1, x_2 = 1, x_3 = 1) &= P(y = 1|x_1 = 1) \times P(y = 1|x_2 = 1) \times P(y = 1|x_3 = 1) \\
&= 0.5 \times 0.67 \times 0.67;
\end{aligned}$$

Following Step 3 of the above procedure, the following implication is made:

$$P(y = 1|x_1 = 1, x_2 = 1, x_3 = 1) > P(y = 0|x_1 = 1, x_2 = 1, x_3 = 1) \rightarrow y = 1;$$

Therefore, the test instance is assigned 1 as the value of y .

Distance measures have been popularly used in the K Nearest Neighbour (KNN) algorithm. The learning outcome of this algorithm is to assign the test instance the class which is the most commonly occurring from the k instances chosen from the training set. The learning strategy of this algorithm is in the following procedure:

Step 1: Choosing a method of measuring the distance between two points, e.g. Euclidean Distance.

Step 2: Determining the value of K , i.e. the number of training instances being selected.

Step 3: Finding the k instances (data points) that are closest to the given test instance.

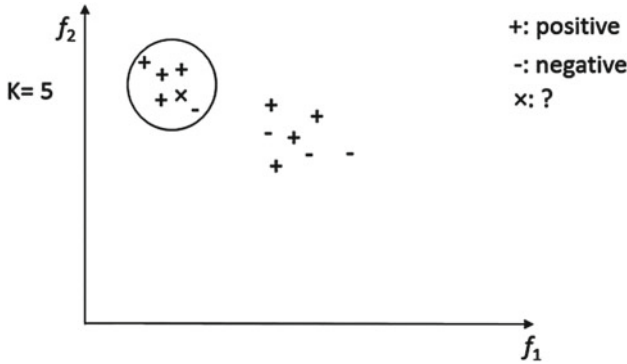


Fig. 2.1 K Nearest Neighbour for two class classification [1]

Step 4: Identifying the majority class that is most commonly occurring from the k instances chosen at step 2.

Step 5: Assigning the test instance the majority class identified at step 3.

An example is shown in Fig. 2.1 for illustrating the procedure of the KNN algorithm.

In the above example, the chosen value of K is 5 and there are two possible classes (positive or negative) towards classifying the test instance. As shown in Fig. 2.1, the five instances, which are closest to the test instance in terms of Euclidean Distance, are surrounded within a circle. In particular, there are four positive instances and one negative instance among the chosen ones. Therefore, the positive class is finally selected through majority voting towards classifying the above test instance.

On the basis of the above description, both NB and KNN would lead to deterministic classification. In other words, the repetitive conduct of the same experiment does not lead to variance of classification results, while both the training and test sets are fixed. The phenomenon of deterministic classification results indicates that heuristic learning generally leads to low variance but could result in high bias. More detailed analysis of bias and variance will be provided in Sect. 2.6, and some issues of heuristic learning in the big data era will be identified in Sect. 2.7.

2.3 Single-Task Learning

Single-task learning generally means that the learning outcome is single. In other words, the learning is aimed at predicting the value of a particular output attribute. In this context, there must be only a single test set involved in the testing stage. Also, both training and test sets can only contain one output attribute.

In traditional machine learning, classification is typically undertaken as a single task, which is simply aimed at classifying an instance into one of two or more cate-

gories. In practice, this kind of classification has been involved in broad application areas, such as medical diagnosis [2], pattern recognition [3], sentiment analysis [4–6] and ontology engineering [7].

On the other hand, traditional learning approaches can be categorized into single learning and ensemble learning. The former type involves learning of a single classifier whereas the latter type involves learning of an ensemble classifier. However, both types of classifiers are typically used for predicting the value of a single output attribute, so both single learning and ensemble learning would be considered to belong to single-task learning, unless there are multiple data sets involved in the testing stage or the data set being dealt with contains more than one output attribute.

In the era of big data, due to the increase in the complexity of data characteristics, single-task learning would be much limited against effective processing of big data, due to some issues. More details on these issues will be described in Sect. 2.7.

2.4 Discriminative Learning

Discriminative learning generally means learning to discriminate one class from other classes towards uniquely classifying an unseen instance. In other words, discriminative learning approaches work based on the assumption that different classes are mutually exclusive. In this context, each instance is considered to be clear-cut and thus can only belong to one class. Also, it is typical to force all instances to be classified, i.e. it is not acceptable to leave an instance unclassified.

As identified in [8], the assumption of mutual exclusion among different classes does not always hold in real applications. For example, a movie on war can belong to both military and history, due to the fact that this type of movies tells a real story that involves soldiers and that happened in the past. Also, the same book may be used by students from different departments and thus can belong to different subjects.

On the other hand, while different classes are truly mutually exclusive, instances may be complex leading to difficulty in classifying each of such instances to one category only. For example, in handwritten digits recognition, handwritten ‘4’ and ‘9’ are very similar and sometimes hard to distinguish. From this point of view, it is not appropriate to assume each instance to be clear-cut.

Moreover, a real data set is dynamically grown in practice. In this context, if a classifier is trained using a data set with a certain number of predefined classes, then it is likely to result in inaccurate classifications when the data set is updated [9, 10]. In fact, it is very likely to occur in practice that a data set is initially assigned a number of class labels by domain experts on the basis of their incomplete knowledge, but the data set is later provided with extra labels following the gain of new knowledge.

In the above case, it is not appropriate to force all instances to be classified since it is possible that an instance does not belong to any one of the initially assigned classes. The above case also indicates that the learning of a classifier must be redone once the class labels of the data set are updated, i.e. the classifier is poor in maintainability, re-usability, extendability and flexibility and thus not acceptable in real applications [8].

In order to address the above issues, multi-label classification has recently been proposed. The term multi-label generally means that an instance is assigned multiple class labels jointly or separately, which is the main difference to the term single-label meaning that an instance is assigned a single class label only. Details can be found in [11–14].

There are three typical ways of dealing with multi-label classification problems referred to as PT3, PT4 and PT5 respectively, as reviewed in [11].

PT3 is designed to enable that a class consists of multiple labels as illustrated in Table 2.2. For example, two classes A and B can make up three labels: A , B and $A \wedge B$. PT4 is designed to do the labelling on the same data set separately regarding each of the predefined labels as illustrated in Tables 2.3 and 2.4.

In addition, PT5 is aimed at uncertainty handling. In other words, it is not certain to which class label an instance should belong, so the instance is assigned all the possible labels and is treated as several different instances that have the same inputs but different class labels assigned. An illustrative example is given in Table 2.5: both instances (3 and 4) appear twice with two different labels (A and B) respectively, which would be treated as four different instances (two assigned A and the other two assigned B) in the process of learning.

In the big data era, all of the above approaches for mutli-label classification would lead to negative impacts on learning of classifiers. More details on these impacts will be given in Sect. 2.7.

Table 2.2 Example of PT3 [8]

Instance ID	Class
1	A
2	B
3	$A \wedge B$
4	$A \wedge B$

Table 2.3 Example of PT4 on Label A [8]

Instance ID	Class
1	A
2	$\neg A$
3	A
4	A

Table 2.4 Example of PT4 on Label B [8]

Instance ID	Class
1	$\neg B$
2	B
3	B
4	B

Table 2.5 Example of PT5 [8]

Instance ID	Class
1	A
2	B
3	A
3	B
4	A
4	B

2.5 Random Data Partitioning

In machine learning, there are several ways of data partitioning for experimentation. The most popular ways are typically referred to as training/test partitioning or cross-validation [15–17].

The training/test partitioning approach typically involves the partitioning of data into a training set and a test set in a specific ratio, e.g. 70% of the data is used as the training set and 30% of the data is used as the test set. This data partitioning can be done randomly or in a fixed way (e.g. the first 70% of the instances in the dataset are assigned to training set and the rest to the test set). The fixed way is typically avoided (except when order matters) as it may introduce systematic differences between the training set and the test set, which leads to sample representativeness issues. To avoid such systematic differences, the random assignment of instances into training and test sets is typically used.

As mentioned in Chap. 1, cross-validation is conducted by partitioning a data set into n folds (or subsets), followed by an iterative process of combining the folds into different training and test sets. Due to the case that cross-validation is generally more expensive in terms of computational cost, the training/test partitioning approach is often used instead. Also, there have been some new perspectives of using the above two ways of data partitioning identified in [18]. In other words, cross-validation and the training/test partitioning approach would be used for different purposes in machine learning experimentation, and more details on the differences will be presented and discussed in Chap. 8.

In the big data era, random partitioning of data could lead to more serious issues that affect the performance of learning algorithms. More details on these issues will be given in Sect. 2.7.

2.6 General Issues

In traditional machine learning, prediction errors can be caused by bias and variance. Bias generally means errors originated from heuristics of learning algorithms and variance means errors originated from random processing of data. In terms of learning

performance, heuristic learning introduced in Sect. 2.2 generally leads to high bias but low variance, and random partitioning of data introduced in Sect. 2.5 generally leads to low bias but high variance. From this point of view, it has become necessary to manage the trade-off between bias and variance, towards optimizing the performance of learning.

A principal problem of machine learning is referred to as overfitting [19], which essentially means that a model performs to a high level of accuracy on training data but to a low level of accuracy on test data. It was claimed traditionally [20] that the cause of overfitting would be due to the bias originated from the heuristics of the employed learning algorithm leading to the poor generality of the learned model. However, we argue that the overfitting problem is also caused due to the sample representativeness issue, i.e. a huge dissimilarity between training and test sets.

For example, as illustrated in Fig. 2.2, the hypothesis space is covered by the whole data set, and the training space results from the distribution of the training instances, which is a small subspace of the hypothesis space.

In this case, if the test instances are all distributed inside the training space, which means that the test instances are highly similar to the training instances, then it would be very likely that the model would perform to a high level of classification accuracy. However, if most test instances are located far away from the training space, which means that the test instances are mostly dissimilar to these training instances, then it would be very unlikely for the model to perform to a high level of classification accuracy.

In Sect. 2.7, we will analyze further the impact of the sample representativeness issue on learning performance in the big data era.

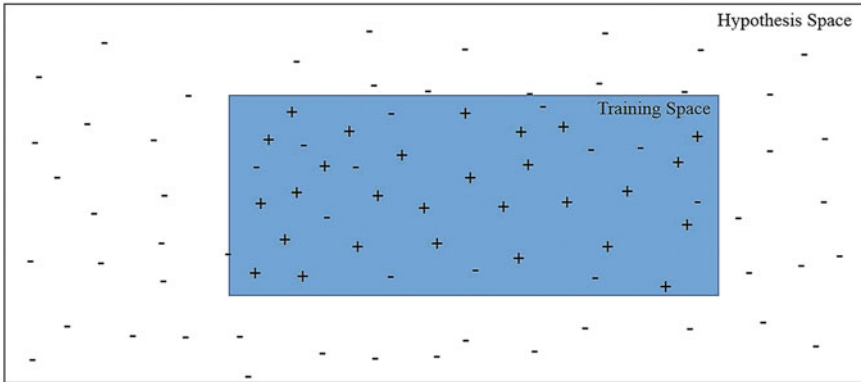


Fig. 2.2 Illustration of Overfitting (NB: “+” indicates training instance and “-” indicates test instance)

2.7 Impacts from Big Data

In the big data era, learning tasks have become increasingly more complex. In this context, traditional machine learning has been too shallow to deal with big data sufficiently. In this section, we analyze how the traditional learning approaches presented in Sects. 2.1–2.5 would be constrained in the context of big data processing.

In terms of supervised learning, as mentioned in Sect. 2.1, the data used in a learning task must be fully labelled by experts. However, due to the vast and rapid increase in the size of data, it has become an overwhelming task to have all instances labelled by experts. From this point of view, it has been necessary to turn supervised learning into semi-supervised learning, towards involving machine based labelling instead of using only expert based labelling. In other words, a small part of a large data set is labelled by experts, and the rest of the data set is labelled by using learning algorithms on the basis of the experts labelled data subset. In Chap. 3, we will present how to achieve semi-supervised Learning through machine based labelling.

In terms of heuristic learning, as mentioned in Sect. 2.2, it could lead to high bias and low variance in terms of learning performance, when both training and test sets are fixed. However, in the big data era, the characteristics of data could be changed quickly and significantly. In this context, high bias would lead to the case that a learning approach is highly suitable for dealing with the original data set but becomes very unsuitable later due to the significant change of data characteristics (high variability of data). From this point of view, it has become necessary to turn heuristic learning into semi-heuristic learning towards achieving the trade-off between bias and variance. In Chap. 4, we will present how to achieve semi-heuristic learning through nature inspired techniques. Also, we will show in Chap. 8 how greatly the high variability of data can impact on the performance of heuristic learning approaches.

In terms of single-task learning, as mentioned in Sect. 2.3, both training and test sets can only have one output attribute.

In the big data era, classification can be from different perspectives, which indicates that it has become normal to learn from the same feature set for different classification tasks. For example, in the context of student classification, we can classify a student in terms of their study mode (full-time or part-time), their learning performance (good or bad) and their degree level (undergraduate or postgraduate), on the basis of the same feature set. From this point of view, it has become necessary to turn single-task learning into multi-task learning.

Also, single task learning requires that there must be only a single test set involved in the testing stage. In the context of big data processing, due to a large size of data, it has been necessary to decompose a data set into different subsets [10], such that learning from different subsets can be done in parallel. From this point of view, each subset is divided into a training subset and a test subset for a particular learning task, which again indicates the necessity of turning single-task learning into multi-task learning.

In Chap. 5, we will present how to achieve multi-task learning through using fuzzy approaches.

In terms of discriminative learning, as mentioned in Sect. 2.4, it is based on the assumption that different classes are mutually exclusive. Since the assumption does not always hold for practical applications, multi-label learning approaches (PT3, PT4 and PT5) have been proposed for addressing this issue. As argued in [8], these approaches are still aimed at dealing with classification problems in a discriminative way, i.e. the aim is still to learn a classifier that provides a unique output, in terms of the class to which an instance belongs.

In the big data era, PT3 may result in a massive number of classes, i.e. $2^n - 1$, where n is the number of class labels. PT4 may result in class imbalance. For example, a balanced data set involves three classes A, B and C, and thus the frequency ($\frac{1}{3}$) of class A is far lower than the one ($\frac{2}{3}$) of class $\neg A$ (i.e. B+C). PT5 may result in a massive size of training sample leading to high computational complexity. From software engineering perspectives, PT3 may result in high coupling, while different class labels that are not correlated are merged into a new class. Coupling generally refers to the degree of interdependence between different parts [21]. Similarly, PT4 may result in low cohesion, while different class labels that are correlated get separated. Low cohesion means the degree to which the parts of a whole link together is lower [21], and thus failing to identify the correlations between different classes.

On the basis of the above argumentation on discriminative learning, it has been necessary to address further the issue of mutual exclusion among classes. In particular, we will present in Chap. 5 how to adopt generative learning through fuzzy approaches towards dealing with the above issue.

In terms of random data partitioning, as mentioned in Sect. 2.5, it is typically conducted by selecting training/test instances in a particular probability, e.g. if the distribution between training and test sets needs to be 70/30, then an instance has a 70% chance to be selected as a training one. In the big data era, this way of partitioning is more likely to lead to two major issues: (a) class imbalance and (b) sample representativeness issues.

The first issue of class imbalance [22, 23] is known to affect the performance of many learning algorithms. Randomly partitioning the data, however, can lead to class imbalance in both the training set and the test set, even when there is no imbalance in the overall data set. For example, let us consider a 2-class (e.g. positive and negative class) data set with a balanced distribution of instances across classes, i.e. 50% of the instances belong to the positive class and 50% of the instances belong to the negative class. When the data set is partitioned by selecting training/test instances randomly, it is likely that the class balance of the data set will be broken, which would lead, for example, to more than 50% of the training instances belonging to the positive class and more than 50% of the test instances belonging to the negative class, i.e. the training set has more positive instances than negative ones, while the test set has the opposite situation.

The second issue is about sample representativeness and the fact that the random partitioning may lead to high dissimilarity between training and test instances, as mentioned in Sect. 2.6. In the context of student learning, the training instances are like the revision questions and the test instances are like the exam questions. To

test effectively the performance of student learning, the revision questions should be representative with respect to the learning content covered in the exam questions. The random partitioning of data, however, can result in the case that the training instances are dissimilar to the test instances, which corresponds to the situation that students are tested on what they have not yet learned. Such a situation not only leads to a poor performance, but also to a poor judgment of the learner capability. Thus, in the context of predictive modelling, some algorithms may be judged as not being suitable for a particular problem due to a poor performance, when in reality the poor results are not due to the algorithm, but to the representativeness of the training sample.

In order to address the above two issues, it has been necessary to turn random partitioning into semi-random partitioning. In Chap. 6, we will present a multi-granularity framework for semi-random partitioning of data, which focuses on dealing with the class imbalance issue and provides a brief proposal towards dealing effectively with the sample representativeness issue.

References

1. H. Liu, A. Gegov, and M. Cocca. 2016. Nature and biology inspired approach of classification towards reduction of bias in machine learning. In *International Conference on Machine Learning and Cybernetics*, Jeju Island, South Korea, 10-13 July 2016, pp. 588–593.
2. Cendrowska, J. 1987. Prism: an algorithm for inducing modular rules. *International Journal of Man-Machine Studies* 27: 349–370.
3. Z. Teng, F. Ren, and S. Kuroiwa. 2007. Emotion recognition from text based on the rough set theory and the support vector machines. In *International Conference on Natural Language Processing and Knowledge Engineering*, Beijing, China, 30 August- 1 September 2007, pp. 36–41.
4. K. Reynolds, A. Kontostathis, and L. Edwards. 2011. Using machine learning to detect cyberbullying. In *Proceedings of the 10th International Conference on Machine Learning and Applications*, December 2011, pp. 241–244.
5. Tripathy, A., A. Agrawal, and S.K. Rath. 2015. Classification of sentimental reviews using machine learning techniques. *Procedia Computer Science* 57: 821–829.
6. H. Liu and M. Cocca. 2017. Fuzzy rule based systems for interpretable sentiment analysis. In *International Conference on Advanced Computational Intelligence*, Doha, Qatar, 4–6 February 2017, pp. 129–136.
7. Pedrycz, W., and S.M. Chen. 2016. *Sentiment Analysis and Ontology Engineering: An Environment of Computational Intelligence*. Heidelberg: Springer.
8. H. Liu, M. Cocca, A. Mohasseb, and M. Bader. 2017. Transformation of discriminative single-task classification into generative multi-task classification in machine learning context. In *International Conference on Advanced Computational Intelligence*, Doha, Qatar, 4–6 February 2017, pp. 66–73.
9. Wang, X., and Y. He. 2016. Learning from uncertainty for big data: Future analytical challenges and strategies. *IEEE Systems, Man and Cybernetics Magazine* 2 (2): 26–32.
10. Suthaharan, S. 2014. Big data classification: problems and challenges in network intrusion prediction with machine learning. *ACM SIGMETRICS Performance Evaluation Review* 41 (4): 70–73.
11. Tsoumakas, G., and I. Katakis. 2007. Multi-label classification: an overview. *International Journal of Data Warehousing and Mining* 3 (3): 1–13.

12. I. K. G. Tsoumakas and I. Vlahavas. 2010. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook*. Springer 2010, pp. 667–685.
13. Boutell, M.R., X.S.J. Luo, and C.M. Brown. 2004. Learning multi-label scene classification. *Pattern Recognition* 37: 1757–1771.
14. Zhang, M., and H. Zhou. 2014. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering* 26 (8): 1819–1837.
15. R. Kohavi. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*. San Mateo, CA: Morgan Kaufmann, pp. 1137–1143.
16. Geisser, S. 1993. *Predictive Inference*. New York: Chapman and Hall.
17. Devijver, P.A. 1982. *Pattern Recognition: A Statistical Approach*. London: Prentice-Hall.
18. Liu, H., A. Gegov, and M. Cocea. 2017. Unified framework for control of machine learning tasks towards effective and efficient processing of big data. In *Data Science and Big Data: An Environment of Computational Intelligence*, pp. 123–140. Switzerland: Springer.
19. Liu, H., A. Gegov, and M. Cocea. 2016. *Rule Based Systems for Big Data: A Machine Learning Approach*. Switzerland: Springer.
20. Mitchell, T. 1997. *Machine Learning*. New York: McGraw Hill.
21. T. C. Lethbridge and R. Laganire. 2005. *Object Oriented Software Engineering: Practical Software Development using UML and Java (2nd)*. Maidenhead: McGraw-Hill Education.
22. Longadge, R., S.S. Dongre, and L. Malik. 2013. Class imbalance problem in data mining: Review. *International Journal of Computer Science and Network* 2 (1): 83–87.
23. Ali, A., S.M. Shamsuddin, and A.L. Ralescu. 2015. Classification with class imbalance problem: a review. *International Journal of Advanced Soft Computing Applications* 7 (3): 176–204.

Granular Computing Based Machine Learning
A Big Data Processing Approach

Liu, H.; Cocea, M.

2018, XV, 113 p. 27 illus., 19 illus. in color., Hardcover

ISBN: 978-3-319-70057-1