NUMERICAL MACHINE LEARNING

Zhiyuan Wang
Sayed Ameenuddin Irfan
Christopher Teoh
Priyanka Hriday Bhoyar

Bentham Books

Numerical Machine Learning

Authored by

Zhiyuan Wang
DigiPen Institute of Technology Singapore
Singapore

National University of Singapore
Singapore

Sayed Ameenuddin Irfan
DigiPen Institute of Technology Singapore
Singapore

Christopher Teoh
DigiPen Institute of Technology Singapore
Singapore

&

Priyanka Hriday Bhoyar
DigiPen Institute of Technology Singapore
Singapore

Numerical Machine Learning

Authors: Zhiyuan Wang, Sayed Ameenuddin Irfan, Christopher Teoh & Priyanka Hriday Bhoyar

ISBN (Online): 978-981-5136-98-2

ISBN (Print): 978-981-5136-99-9

ISBN (Paperback): 978-981-5165-00-5

First published in 2023.

need for a court order if at any point you breach any terms of this License Agreement. In no event will any delay or failure by Bentham Science Publishers in enforcing your compliance with this License Agreement constitute a waiver of any of its rights.

3. You acknowledge that you have read this License Agreement, and agree to be bound by its terms and conditions. To the extent that any other terms and conditions presented on any website of Bentham Science Publishers conflict with, or are inconsistent with, the terms and conditions set out in this License Agreement, you acknowledge that the terms and conditions set out in this License Agreement shall prevail.

Bentham Science Publishers Pte. Ltd.
80 Robinson Road #02-00
Singapore 068898
Singapore
Email: subscriptions@benthamscience.net

BENTHAM SCIENCE

CONTENTS

PREFACE

In recent years, machine learning has become increasingly popular and pervasive, with applications ranging from self-driving cars and facial recognition to personalized website recommendations and stock market forecasting. The increased availability of data and advancements in computer power have made it possible to apply machine learning algorithms to a vast array of problems with impressive outcomes. Machine learning is currently utilized in a variety of areas, including banking, healthcare, marketing, and manufacturing, and it is anticipated that it will continue to play a significant role in the development of new technologies in the future. Consequently, machine learning has emerged as an essential subject of study for people interested in data science, artificial intelligence, and related fields. As machine learning continues to evolve and expand its reach, researchers and practitioners are constantly developing new techniques and algorithms to address specific challenges or improve upon existing methods. In this ever-changing landscape, it is crucial for those working in the field to stay up-to-date with the latest advancements and trends. This includes not only mastering the fundamental concepts and algorithms, but also understanding how to adapt and apply them in novel ways to solve real-world problems. By embracing the interdisciplinary nature of machine learning, and collaborating with experts from diverse fields, we can accelerate the development of innovative solutions that have the potential to transform industries, enhance the quality of life, and create a more sustainable future for all.

From our experiences of teaching machine learning using various textbooks, we have noticed that there tends to be a strong emphasis on abstract mathematics when discussing the theories of machine learning algorithms. On the other hand, in the application of machine learning, it usually straightaway goes to import off-the-shelf libraries such as scikit-learn, TensorFlow, Keras, and PyTorch. The disconnect between abstract mathematical theories and practical application creates a gap in understanding. This book bridges the gap using numerical examples with small datasets and simple Python codes to provide a complete walkthrough of the underlying mathematical steps of machine learning algorithms. By working through concrete examples step by step, readers/students can develop a well-rounded understanding of these algorithms, gain a more in-depth knowledge of how mathematics relates to the implementation and performance of the algorithms, and be better equipped to apply them to practical problems.

Beginning with an introduction to machine learning in Chapter 1, the remaining chapters of the book cover seven commonly used machine learning algorithms and techniques, including both supervised and unsupervised learning, as well as both linear and nonlinear models. The book requires some prerequisite knowledge of basic probability and statistics, linear algebra, calculus, and Python programming. The book is intended for university students studying machine learning and is used as our primary teaching material for the "Introduction to Machine Learning" module at DigiPen Institute of Technology Singapore.

In conclusion, we would like to acknowledge Mr. Tan Chek Ming (Managing Director), Prof. Prasanna Ghali (Provost), Ms. Caroline Tan (Deputy Director), Ms. Angela Tay (Senior Manager), and all at DigiPen Institute of Technology Singapore, for their consistent support and help. We also wish to thank a number of our students (including Nelson Ng, Rhonda McGladdery, Farhan Fadzil, Lim Li Jia, Musa Ahmad Dahlan, Jeremy Yap, and Seah Jue Chen) for their diligence in spotting several typographical errors during their course of studies. Also, it has been a delight working with Bentham's professional editorial and production staff. We particularly thank Noor Ul Ain Khan, Humaira Hashmi, and Obaid Sadiq for their consistent, timely, and kind support throughout the development of this book. Furthermore, we extend our heartfelt appreciation to our families (including Xiaoyue Cui, Muyuan Wang, Safura Tazeen, Khasim BI, Shirleen Chow, Adler Teoh, Hriday Bhoyar, Swati Kolkhede, and all) for their unwavering encouragement throughout the creation of this book. We dedicate this book to them. The first author, Zhiyuan Wang, would also like to convey special thanks and appreciation to his Ph.D. advisors, Prof. Zhe Wu, Prof. Xiaonan Wang, and Prof. Gade Pandu Rangaiah from the National University of Singapore. Although they were not involved in this book, Zhiyuan deeply cherishes their sincere and invaluable guidance in his Ph.D. journey, which has helped him become a better researcher and educator.

Despite our best efforts to ensure the accuracy of the content within this book, errors may inadvertently persist. If you come across any inaccuracies or omissions, we kindly request that you bring them to our attention by emailing us at wangzhiyuan@u.nus.edu. We are committed to rectifying such oversights in future editions and will post corrections on our shared folder in Google Drive: https://drive.google.com/drive/folders/1FqJvo4ZPazNbEH_GlHFoodqvegnQmHcn?usp=share_link

Zhiyuan Wang
DigiPen Institute of Technology Singapore
Singapore
National University of Singapore
Singapore

Sayed Ameenuddin Irfan
DigiPen Institute of Technology Singapore
Singapore

Christopher Teoh
DigiPen Institute of Technology Singapore
Singapore

&

Priyanka Hriday Bhoyar
DigiPen Institute of Technology Singapore
Singapore

CHAPTER 1

Introduction to Machine Learning

Abstract: Machine learning, a rapidly growing subfield of computer science, has had a significant impact on many industries and our lives. This chapter discusses the brief history of machine learning, its widespread adoption as a de facto feature, and fundamental concepts such as supervised and unsupervised learning, regression and classification, and underfitting and overfitting. We also emphasize the importance of understanding machine learning through numerical examples, which can bridge the gap between abstract mathematical theories and practical applications of machine learning algorithms. By developing a strong foundation in machine learning, readers/students can harness its potential to address challenges and opportunities across diverse sectors.

Keywords: Numerical Examples, Machine Learning History, Supervised Learning, Unsupervised Learning, Regression, Classification, Underfitting, Overfitting

1.1. BRIEF HISTORY OF MACHINE LEARNING

Machine learning is a subfield of computer science that involves the creation of algorithms that can learn from data and make predictions. It has a long and rich history [1], with roots dating back to the 1950s when the field of artificial intelligence was founded. This field focused on developing machines that could perform tasks that typically require human-like intelligence, such as recognizing patterns, learning from experience, and making decisions. The first machine learning algorithms were developed in the 1960s, including decision tree and nearest neighbor algorithms. The 1980s saw the rapid growth of the field with the development of algorithms such as artificial neural network and support vector machine. These algorithms were applied to a wide range of applications in the 1990s, including natural language processing, computer vision, and speech recognition. In the 2000s, the field continued to evolve with the development of new algorithms, such as gradient boosting, and the increasing use of machine learning in industries such as finance and healthcare. The 2010s saw the widespread adoption of machine learning, aided by the advent of big data and the development of powerful graphics processing units (GPU) that could be used to train large and complex machine learning models. The subfield of deep learning [2], which typically involves the use of multi-layered neural networks, became particularly popular and found application across a diverse range of domains. Today, machine learning is a rapidly growing field that is currently being applied in various sectors.

Zhiyuan Wang, Sayed Ameenuddin Irfan, Christopher Teoh & Priyanka Hriday Bhoyar

It has the potential to revolutionize many industries and has already had a significant societal impact.

1.2. MACHINE LEARNING AS A DE FACTO FEATURE

Machine learning is expected to be a transformative technology over the next two decades due to several factors. One key factor is the increasing availability of data, which is expected to continue to grow significantly in the coming years. As machine learning algorithms are particularly well suited for analyzing and making sense of large amounts of data, this will create new opportunities for their application in a variety of fields, including but not limited to healthcare, finance, transportation, education, manufacturing, and beyond. In these and other areas, machine learning has been adopted to automate some tasks that are currently performed by humans, freeing up humans to focus on more creative and high-level work [3].

In addition to automation, machine learning algorithms can be used to improve decision-making by analyzing large amounts of data and making predictions or recommendations based on that data. This can be particularly useful in fields such as finance, where machine learning can be used to identify patterns and trends that can inform investment decisions, or in healthcare, where machine learning can be used to predict patient outcomes and identify potential health risks, or in semiconductor manufacturing, where machine learning can be employed to detect defects and analyze their causes in real-time. By providing valuable insights and recommendations based on data analysis, machine learning has the potential to enhance the efficiency and effectiveness of decision-making in a wide range of fields.

Another key benefit of machine learning is its ability to enhance personalization by tailoring products and services to individual preferences and behaviors. For example, machine learning can be used to recommend products or content to users based on their past behavior, or to tailor advertising to specific audiences. By providing personalized experiences, machine learning has the potential to improve customer satisfaction and engagement.

Overall, machine learning is expected to have a significant impact in a wide range of fields over the next two decades, influencing many aspects of our lives. Its ability to automate tasks, improve decision-making, and enhance personalization make it a technology with the potential to revolutionize industries and change the way we live and work.

1.3. SUPERVISED AND UNSUPERVISED

Supervised and unsupervised learning are two prominent types of algorithms in machine learning [4]. In supervised learning, a model is trained using labeled data, which includes the correct output for each instance in the training set. The model generates predictions based on this labeled data, enabling it to make accurate predictions for new, previously unseen examples. Some common supervised learning tasks include regression, which aims to predict a continuous value, and classification, which focuses on predicting a categorical label. Conversely, unsupervised learning involves training a model with unlabeled data, meaning the correct output is not provided. In this case, the model must independently identify patterns and relationships within the data. Examples of unsupervised learning tasks encompass clustering, where the objective is to group similar examples, and dimensionality reduction, where the goal is to decrease the number of features in the data while preserving as much relevant information as possible.

1.4. REGRESSION AND CLASSIFICATION

In machine learning, regression and classification are two types of supervised learning, in which a model is trained on labeled data to make predictions about new, unseen examples. In regression, the model is used to predict a continuous value, such as a price or probability. For example, a regression model might be used to predict the price of a house based on features such as its size, number of bedrooms, and location. On the other hand, classification involves predicting a categorical value, such as a class label. For example, a classification model might be used to predict whether an email is spam or not, or to recognize the type of object in an image.

Both regression and classification are widely used in many fields and have a broad range of applications. In addition to the examples mentioned earlier, regression can be applied in finance to predict stock prices, in healthcare to predict patient outcomes, in meteorology to predict weather patterns, and in electric vehicle industry to predict charging demand [5]. Classification, on the other hand, is used in a wide range of applications, such as natural language processing, where it is used to classify text into different categories, and fraud detection, where it is used to classify transactions as legitimate or fraudulent. Despite their differences, regression and classification share many similarities and are both essential tools in the field of machine learning. By understanding both, we can select the most appropriate method for a specific problem and achieve more accurate predictions.

CHAPTER 2

Linear Regression

Abstract: In this chapter, we delve into linear regression, a fundamental machine learning algorithm for predicting numerical values. While maintaining a concise overview of the mathematical theories, we prioritize an accessible approach by focusing on a concrete numerical example with a small dataset for predicting house sale prices. Through a step-by-step walkthrough, we illustrate the inner workings of linear regression and demonstrate its practical implementation. Additionally, we offer sample codes and a comparison with the linear regression model from scikit-learn to reinforce understanding. Upon completing this chapter, readers will gain a comprehensive understanding of linear regression's inner workings and its relationship to algorithm implementation and performance, and be better prepared to apply it to real-world projects.

Keywords: Linear Regression, Numerical Example, Small Dataset, Housing Price Prediction, Scikit-Learn

2.1. INTRODUCTION TO LINEAR REGRESSION

Linear regression is a supervised machine learning algorithm that aims to determine the best-fit linear line between a dependent variable and one or more independent variables. It typically carries out regression tasks. It is one of the easiest, most well-understood, and most popular algorithms in many machine learning applications [1, 2]. It can be employed to predict the values of continuous numerical variables such as salary, sales revenue, dividend yield, greenhouse gas emission, and house price, to name a few.

Despite its simplicity, linear regression remains a powerful tool in the field of machine learning, providing a strong foundation for understanding the underlying input-output relationships between variables. It serves as an excellent starting point for beginners in the field, offering a straightforward and interpretable approach to modeling. Moreover, linear regression can act as a benchmark for evaluating the performance of more complex algorithms, allowing practitioners to gauge the effectiveness of their chosen models. While linear regression may not always be the most advanced or accurate method for every situation, its ease of use, interpretability, and versatility continue to make it a valuable asset in a variety of real-world applications and industries.

There are several fundamental assumptions associated with linear regression [3, 4]. Firstly, it is assumed that the dependent variable is linearly correlated to the

Zhiyuan Wang, Sayed Ameenuddin Irfan, Christopher Teoh & Priyanka Hriday Bhoyar

independent variable(s). Secondly, when there is more than one independent variable, no correlation should exist between the independent variables (*i.e.*, no multicollinearity). Thirdly, the errors between the true values and predicted values by the linear regression model should approximately conform to a normal distribution, with most having errors close to 0. Fourthly, the spread of the errors (*i.e.*, the variance of the errors) ought to be constant along the values of the dependent variable. This is technically known as homoscedasticity, which can be checked by creating a scatterplot of errors versus the dependent variable.

2.2. MATHEMATICS OF LINEAR REGRESSION

The mathematics of linear regression starts from a simple linear equation, shown in Equation (2.1) and (Fig. **2.1**), where there is only one independent variable X and one dependent variable Y.

$$Y = b + wX \tag{2.1}$$

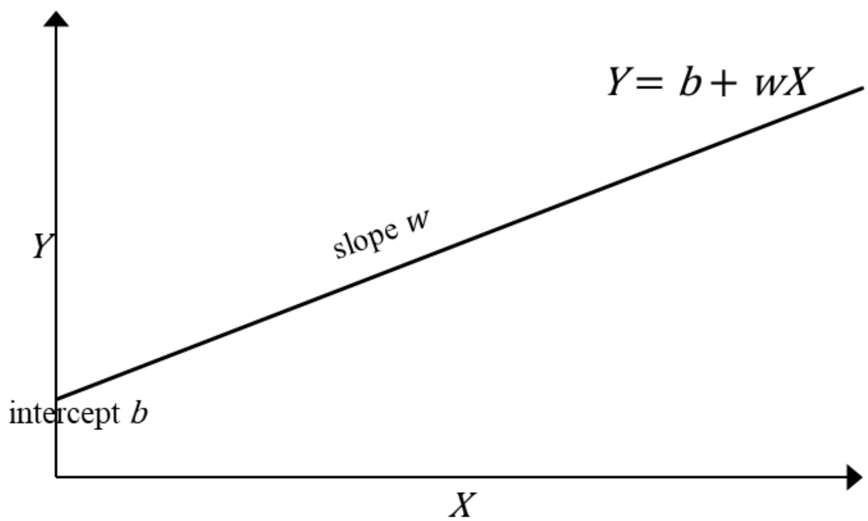

Fig. (2.1). Plot of simple linear equation $Y = b + wX$.

Variable X has an associated coefficient w, which is often used interchangeably with the terms: *weight*, *slope*, or *gradient*. In the context of machine learning, it is most often referred to as weight.

Likewise, b represents the intercept with Y-axis and is often known as *bias* in machine learning. The independent variable X is commonly called *input*, which is used interchangeably with the following terms: *input feature*, *attribute*, *characteristic*, *field*, and *column*. The dependent variable Y is commonly referred to as *output*, *target, class,* and *label*.

In reality, more often than not, we will have more than one independent variable, and Equation (2.1) would have to be updated to a general term Equation (2.2) to cater for this.

$$Y = b + \sum_{j=1}^{n} w_j X_j \qquad \textbf{(2.2)}$$

Here,

Y is the dependent variable
b is the bias
n is the number of input features
w_j is the weight of the j^{th} feature
X_j is input value of the j^{th} feature

The goal of linear regression is to find the best-fit linear equation model that maps the relationship between input X and output Y, in the form of Equation (2.2) with the optimal weights and bias, which produces the least error (synonymously known as loss in machine learning) between the known true output Y values and predicted output Y values by the model.

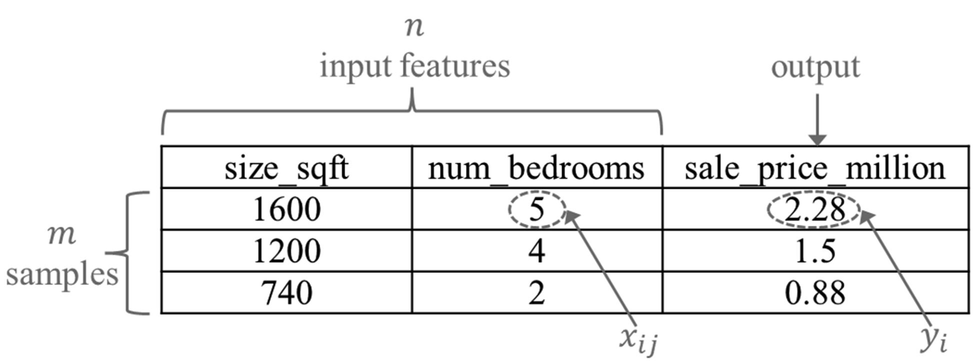

Fig. (2.2). Illustration of notations using an exemplary training dataset.

Let us use lowercase x_i to denote the feature values of the i^{th} sample (out of the total m rows of samples from the training dataset), then x_{ij} will be the value of the j^{th} feature (out of the total n input features) at the i^{th} sample, as shown in Fig.

CHAPTER 3

Regularization

Abstract: This chapter delves into L1 and L2 regularization techniques within the context of linear regression, focusing on minimizing overfitting risks while maintaining a concise presentation of mathematical theories. We explore these techniques through a concrete numerical example with a small dataset for predicting house sale prices, providing a step-by-step walkthrough of the process. To further enhance comprehension, we supply sample codes and draw comparisons with the Lasso and Ridge models implemented in the scikit-learn library. By the end of this chapter, readers will acquire a well-rounded understanding of L1 and L2 regularization in the context of linear regression, their implications on model implementation and performance, and be equipped with the knowledge to apply these methods in practical use.

Keywords: L1 Regularization, L2 Regularization, Linear Regression, Numerical Example, Small Dataset, Housing Price Prediction, Scikit-Learn, Lasso, Ridge

3.1. INTRODUCTION TO L1 AND L2 REGULARIZATION

Regularization is the process of adding an extra penalty to a more complicated model with larger values of weights to prevent overfitting. A problem known as overfitting happens when a machine learning model is made specifically for training datasets and is unable to generalize well to previously unseen datasets. Introducing regularization techniques into machine learning models is essential for achieving better generalization and improved performance on new data. By penalizing overly complex models, regularization helps lead to more accurate and stable predictions. Some popular regularization methods include L1 and L2 regularization, which differ in the way they penalize the model's complexity. Regularization has proven to be a critical component in the development of robust and reliable models, particularly when dealing with high-dimensional data or noisy datasets. It enables practitioners to build more efficient models, capable of adapting to new and diverse situations while reducing the risk of overfitting and maintaining interpretability.

This chapter focuses on L1 and L2 regularization, which are demonstrated in detail by making necessary changes from the original linear regression model discussed in Chapter 2. Bear in mind, however, that L1 and L2 regularization can also be applied to other machine learning models (*e.g.*, logistic regression), as well as deep learning neural networks.

Zhiyuan Wang, Sayed Ameenuddin Irfan, Christopher Teoh & Priyanka Hriday Bhoyar

The names of L1 and L2 regularization come from the corresponding L1 and L2 norms of the weight vector W.

The L1 norm is defined as:

$$\|W\|_1 = \sum_{j=1}^{n} |w_j| = |w_1| + |w_2| + \cdots + |w_n|$$

The L2 norm is defined as:

$$\|W\|_2 = \left(\sum_{j=1}^{n} w_j^2\right)^{1/2} = (w_1^2 + {w_2}^2 + \cdots + {w_n}^2)^{1/2}$$

Here, w_j is the weight of the j^{th} feature, and n is the number of input features.

Note that $\|W\|$, without subscript, is also conventionally used to represent the L2 norm of the weight vector W.

A linear regression model with the L1 regularization is known as Lasso (least absolute shrinkage and selection operator) regression [1, 2], whereas a linear regression model with the L2 regularization is called Ridge regression [3, 4].

3.2. MATHEMATICS OF L1 REGULARIZATION FOR LINEAR REGRESSION

The general equation of linear regression if having more than one independent variable X (*i.e.*, input feature) is as follows:

$$Y = b + \sum_{j=1}^{n} w_j X_j \tag{3.1}$$

Here,

Y is the output
b is the bias
n is the number of input features
w_j is the weight of the j^{th} feature
X_j is input value of the j^{th} feature

The goal of linear regression is to find the best-fit linear equation model that maps the relationship between input X and output Y, in the form of Equation (3.1) with the optimal weights and bias, which produces the least error (synonymously known as loss in machine learning) between the known true output Y values and predicted output Y values by the model.

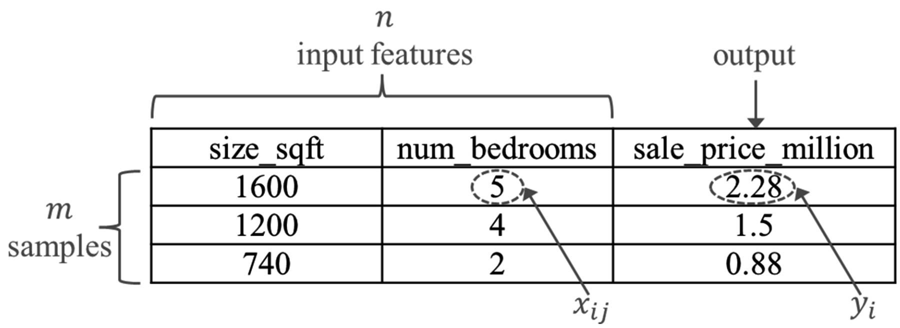

Fig. (3.1). Illustration of notations using an exemplary training dataset.

Let us use lowercase x_i to denote the feature values of the i^{th} sample (out of the total m rows of samples from training dataset), then x_{ij} will be the value of the j^{th} feature (out of the total n input features) at the i^{th} sample, as shown in Fig. (**3.1**). Lowercase y_i is used to represent the known output value (*i.e.*, true output value) of the i^{th} sample, and $\hat{y}_i$ is used to denote the corresponding predicted output value by the linear equation model. Equation (3.1) is then updated to Equation (3.2).

$$\hat{y}_i = b + \sum_{j=1}^{n} w_j x_{ij} \tag{3.2}$$

Here,

$i \in [1, m]$
m is the number of training samples
x_{ij} is the value of the j^{th} feature at the i^{th} sample

b is the bias
n is the number of input features
w_j is the weight of the j^{th} feature
$\hat{y}_i$ is the predicted value for the i^{th} sample

Up to this step, everything is the same as the original linear regression discussed in Chapter 2. The only difference brought by L1 regularization is the change of loss

CHAPTER 4

Logistic Regression

Abstract: This chapter delves into logistic regression, a widely used machine learning algorithm for classification tasks, with a focus on maintaining accessibility by minimizing abstract mathematical concepts. We present a concrete numerical example employing a small dataset to predict the ease of selling houses in the property market, guiding readers through each step of the process. Additionally, we supply sample codes and draw comparisons with the logistic regression model available in the scikit-learn library. Upon completion of this chapter, readers will have gained a comprehensive understanding of the inner workings of logistic regression, its relationship to algorithm implementation and performance, and the knowledge necessary to apply it to practical applications.

Keywords: Logistic Regression, Classification, Numerical Example, Small Dataset, Scikit-Learn

4.1. INTRODUCTION TO LOGISTIC REGRESSION

Logistic regression is a supervised machine learning algorithm for modeling the probability of a discrete output given input features [1, 2]. Despite its name, logistic regression is more of a classification model than a regression model. It is commonly used to model a dichotomous (binary) output, *i.e.*, anything with two possible values/classes/labels, such as true/false, yes/no, 1/0, on/off, good/bad, malignant/ benign, and pass/fail, to name a few. The foundation of logistic regression lies in its ability to model the relationship between input features and a categorical outcome by utilizing the logistic function, also known as the sigmoid function. This function ensures that the predicted probabilities lie within the range of 0 and 1, making it suitable for classification tasks. Logistic regression has gained immense popularity due to its simplicity, interpretability, and efficiency in various real-world applications. Some of these applications include spam filtering, customer churn prediction, medical diagnosis, and credit risk assessment.

Unlike linear regression, logistic regression does not require the assumption of a linear relationship between the independent (X) and dependent (Y) variables. Besides, the errors between the true and predicted outputs need not conform to a normal distribution. Moreover, the spread of the errors (*i.e.*, the variance of the errors) need not be constant along the values of dependent variables; that is, homoscedasticity is not required. However, there are still several essential assumptions for logistic regression [3, 4]. Firstly, when there is more than one independent variable (X), it requires little or no correlation between the independent

Zhiyuan Wang, Sayed Ameenuddin Irfan, Christopher Teoh & Priyanka Hriday Bhoyar

variables (*i.e.*, little or no multicollinearity). Secondly, it assumes that the independent variable(s) have a linear relationship with the logarithm of the odds; odds is just another way of expressing probability (P) and is defined as the ratio of the probability of an event occurring to the probability of an event not occurring (*i.e.*, $\frac{P}{1-P}$). Thirdly, by default, logistic regression is used to solve binary classification problems, requiring the dependent variable (Y) to be dichotomous. On the other hand, it is worth mentioning that with some modifications and improvements like the one-*vs*-rest (OvR) method, logistic regression can be scaled up for solving multi-class classification problems. Nevertheless, multi-class classification is outside the scope of the present chapter as it focuses on binary classification using the logistic regression algorithm.

4.2. MATHEMATICS OF LOGISTIC REGRESSION

Mathematically, the linear regression discussed in Chapter 2 can be upgraded to logistic regression after introducing a sigmoid function for mapping the linear output to probability and employing a different loss function.

In comparison with Equation (2.2) in Chapter 2 for linear regression, the only change made to Equation (4.1) here is to use a variable Z (rather than Y) to represent the linear output, which is just an intermediate result in the process of logistic regression.

$$Z = b + \sum_{j=1}^{n} w_j X_j \tag{4.1}$$

Here,

Z is the intermediate linear output
b is the bias
n is the number of input features
w_j is the weight of the j^{th} feature
X_j is input value of the j^{th} feature

The sigmoid function for mapping the intermediate linear output to probability is defined as Equation (4.2) and plotted in Fig. (**4.1**).

$$Y = \frac{1}{1 + e^{-Z}} \tag{4.2}$$

Here,

Z is the intermediate linear output from the linear Equation (4.1)
Y is the mapped probability

As can be seen from Fig. (**4.1**), the sigmoid function maps the linear output Z into a probability Y that is in the range of 0 to 1. The default threshold is 0.5, meaning that if $Y \geq 0.5$, it will be rounded up to 1 and predicted as class 1; whereas, if $Y < 0.5$, it will be rounded down to 0 and predicted as class 0.

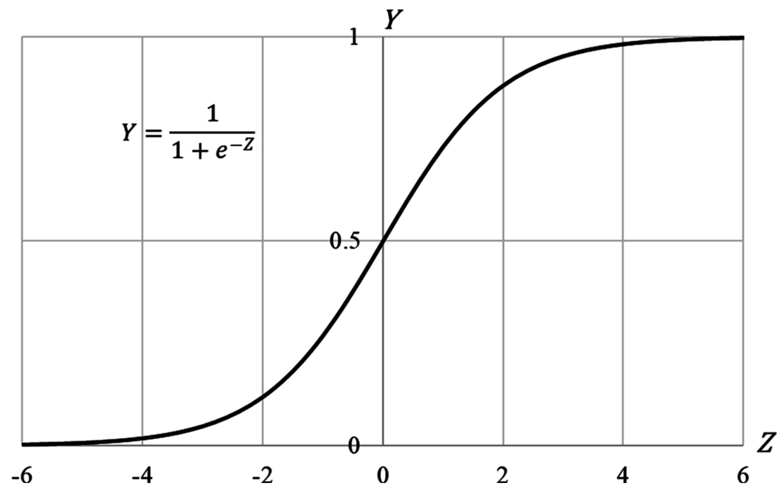

Fig. (4.1). Plot of the sigmoid function.

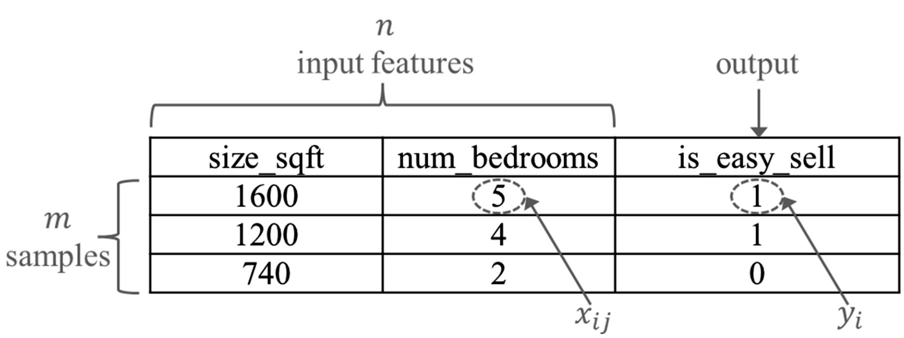

Fig. (4.2). Illustration of notations using an exemplary training dataset.

CHAPTER 5

Decision Tree

Abstract: In this chapter, we explore the concept of decision trees, prioritizing accessibility by minimizing abstract mathematical theories. We examine a concrete numerical example using a small dataset to predict the suitability of playing tennis based on weather conditions, guiding readers through the process step-by-step. Moreover, we provide sample codes and compare them with the decision tree classification model found in the scikit-learn library. Upon completing this chapter, readers will have gained a comprehensive understanding of the inner workings of decision tree machine learning, the relationship between the underlying principles, and the implementation and performance of the algorithm, preparing them to apply their knowledge to practical scenarios.

Keywords: Decision Tree, Classification, Numerical Example, Small Dataset, Scikit-Learn

5.1. INTRODUCTION TO DECISION TREE

A decision tree is a diagrammatic representation of a set of choices and the results of those choices [1]. Decision tree algorithms have become a popular choice for both classification and regression tasks in machine learning due to their inherent advantages. These include their ease of interpretability, as the decision-making process is explicitly laid out in the tree structure, and their efficient training process. Decision trees can handle missing values, automatically select relevant features, and easily manage both numerical and categorical data. Furthermore, they are robust to outliers and noise in the data. Some of the common applications of decision trees include customer segmentation, fraud detection, medical diagnosis, and risk management. Due to their comprehensible nature and ability to visualize complex decision-making processes, decision trees have found widespread adoption in various industries and research fields. Decision tree is a diagram showing the several paths to reach a choice under specific constraints. Each branch symbolizes the decision space, and its leaf nodes are the outcomes. One node, called the root node, is the starting point for the decision tree, and many more branches, including decision nodes and leaf nodes.

For example, as shown in Fig. (**5.1**), consider a situation where one needs to decide whether to go to outdoor sports. The decision tree for this problem may look like this:

Root node: "Should I go to outdoor sports?"

Zhiyuan Wang, Sayed Ameenuddin Irfan, Christopher Teoh & Priyanka Hriday Bhoyar

Decision node: "Is weather good to support outdoor sports?"

If yes, leaf node: "Go to play"

If no, leaf node: "Do not go to play"

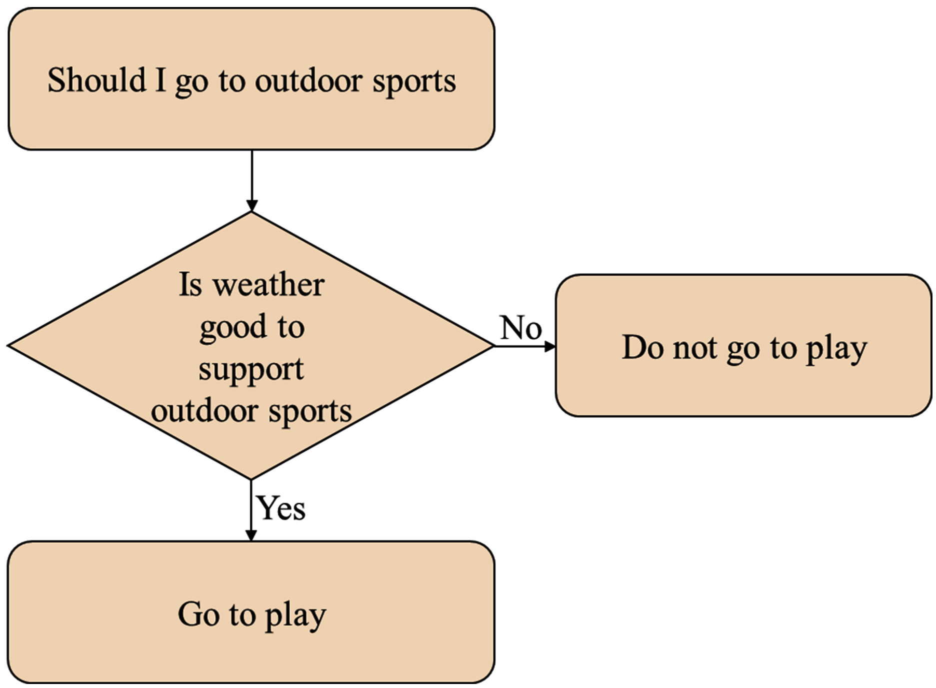

Fig. (5.1). A simple decision tree example.

In this example, the decision node represents the weather condition outside. If the condition is met (*i.e*., the weather is good), the tree leads to the outcome of going to play. If the condition is not met (*i.e*., the weather is not good), the tree leads to the outcome of not going to play.

Now the question might be how to decide which leaf node to select as the root node and decision node. For better decision-making, we use Hunt's algorithm, which helps to give a clear understanding of splitting and choosing the important parameter for root nodes.

5.2. ALGORITHM OF DECISION TREE

In the context of decision tree learning, a heuristic known as Hunt's algorithm is utilized to determine the optimal split for each node in the tree [2]. It is a procedure that iteratively analyzes each feature and the possible value of the feature as a candidate split. Then it chooses the one that yields the most significant increase in information gain (discussed later).

Here is the general process of the Hunt algorithm:

- Calculate the entropy of the current node. Entropy is a measure of the impurity or uncertainty of the data at the node. It is calculated based on the frequencies of the different classes in the data.
- Consider each feature and each possible value of that feature as a candidate split. Calculate the information gain of each candidate split by comparing the entropy of the current node to the entropy of the child nodes that would result from the split.
- Select the split that results in the greatest information gain.
- Repeat the process for each child node, until the desired depth of the tree is reached, or all nodes are pure (*i.e.*, contain only data belonging to a single class).
- Hunt's algorithm is a popular choice for decision tree learning due to its simplicity and efficiency in constructing a decision tree from a dataset. Hunt's algorithm has also served as a foundation for the development of other decision tree algorithms [3], such as ID3 (Iterative Dichotomiser 3), C4.5 (an extension of ID3 that can handle continuous attributes, missing values, and pruning), and CART (Classification and Regression Trees).

As aforementioned, the impurity of a node in a decision tree can be measured using entropy, shown in Equation (5.1). Entropy is calculated based on the frequencies of the different classes in the data. If the data at a node is completely pure, with all data belonging to a single class, then the entropy is zero. On the other hand, if the data is equally divided among all classes, then the entropy is at its maximum.

$$Entropy = \sum_{i=1}^{n} -P_i log_2 P_i \qquad \textbf{(5.1)}$$

Here, P_i is the proportion of class *i* in the node.

Hunt's algorithm recursively splits the data into smaller and smaller subsets until each subset contains data belonging to a single class. At each split, the algorithm

CHAPTER 6

Gradient Boosting

Abstract: In this chapter, we explore gradient boosting, a powerful ensemble machine learning method, for both regression and classification tasks. With a focus on accessibility, we minimize abstract mathematical theories and instead emphasize two concrete numerical examples with small datasets related to predicting house sale prices and ease of selling houses in the property market. By providing a step-by-step walkthrough, we illuminate the inner workings of gradient boosting and offer sample codes and comparisons to the gradient boosting models available in the scikit-learn library. Upon completing this chapter, readers will possess a comprehensive understanding of gradient boosting's mechanics, its connection to the implementation and performance of the algorithm, and be well-prepared to apply it in real-world projects.

Keywords: Gradient Boosting, Ensemble Learning, Regression, Classification, Numerical Example, Small Dataset, Scikit-Learn

6.1. INTRODUCTION TO GRADIENT BOOSTING

Gradient boosting is a robust ensemble machine learning model that sequentially trains a spate of weak learners to produce a more accurate model at the end [1, 2]. A weak learner, generally a rather simple decision tree, is a rudimentary machine learning model with low prediction accuracy but still better than random guessing. As demonstrated in Fig. (**6.1**), the prediction error of the ensemble model is reduced with each new decision tree added and integrated with all the prior decision trees [3]. Gradient boosting is an efficient and accurate algorithm that has been applied to regression and classification problems in many fields, including engineering, healthcare, natural language processing, and computer vision, among others. One of the key strengths of gradient boosting is its ability to leverage the collective knowledge of multiple weak learners, ultimately generating a more robust and accurate model. This is achieved by iteratively focusing on the areas where previous weak learners have failed to make accurate predictions and subsequently improving upon those areas. As a result, gradient boosting has become a popular choice for tackling complex problems and achieving state-of-the-art performance in various applications, even outperforming other ensemble methods, such as random forests in certain contexts. Its versatility and adaptability make gradient boosting a valuable tool in the arsenal of machine learning practitioners and researchers alike.

The rest of this chapter is organized as follows. Section 6.2 presents the mathematics of gradient boosting for regression, followed by the demonstration of a numerical example in detail and code comparison in Section 6.3. Analogously,

Zhiyuan Wang, Sayed Ameenuddin Irfan, Christopher Teoh & Priyanka Hriday Bhoyar

Section 6.4 presents the mathematics of gradient boosting for classification, followed by the demonstration of a numerical example in detail and code comparison in Section 6.5.

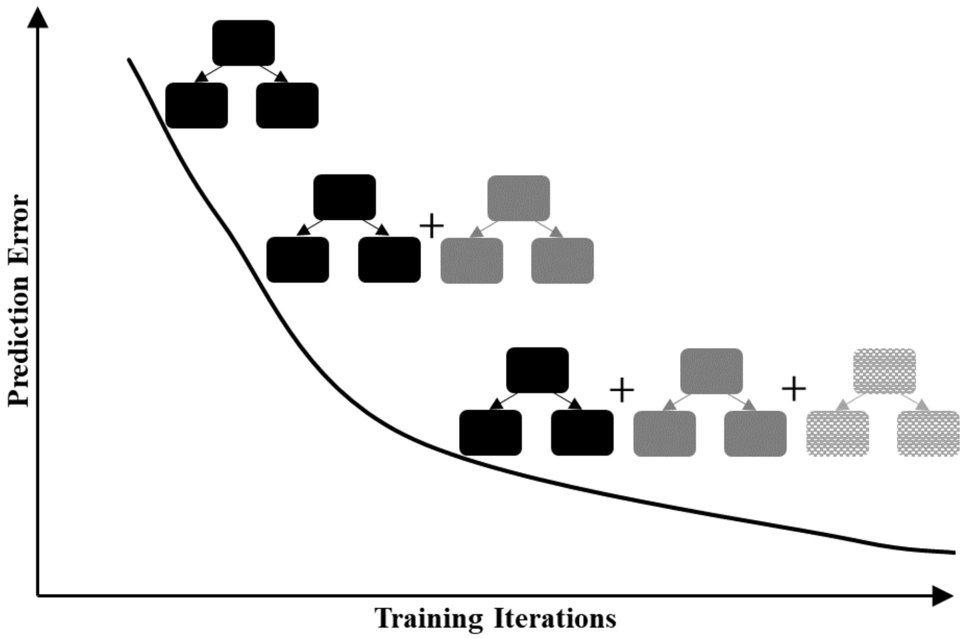

Fig. (6.1). Illustration of gradient boosting.

6.2. MATHEMATICS OF GRADIENT BOOSTING FOR REGRESSION

The basic idea of gradient boosting is to iteratively improve the overall model by fitting the weak learners to the residuals or gradient of the loss function with respect to the previous model's predictions. This process can be viewed as a numerical optimization technique that minimizes the loss function over the training dataset. The algorithm starts by initializing the model with a constant value. The main loop of the algorithm iterates for a predetermined number of iterations (M), and in each iteration, the following four steps are performed. Firstly, the pseudo-residuals are computed by taking the negative gradient of the loss function with respect to the current model's predictions. These pseudo-residuals represent the direction in which the model needs to move to minimize the loss function. Secondly, a weak learner, typically a decision tree, is fit to the pseudo-residuals. The tree is constructed by splitting the input feature space into regions and learning the optimal value for each region to minimize the loss function. Thirdly, the optimal values, denoted by γ, are computed for each region by minimizing the loss function with respect to the previous model's predictions plus the new weak learner's output. Fourthly, the model is updated by adding the weighted output of the new weak learner to the previous model's predictions; the weight here, denoted by α, is a shrinkage parameter that controls the learning rate of gradient boosting. Finally, these four steps are repeated for M iterations, and the final model is a combination of weak learners that can make accurate predictions by collectively minimizing the loss function. Together with the generic pseudocode [4], the mathematics used in gradient boosting for regression is presented in Table **6.1**. Common symbols used throughout the chapter are x_i, denoting the feature values of the i^{th} sample (out of the total n samples from training dataset); y_i and $F(x_i)$, representing the true and predicted output for the i^{th} sample, respectively; m and M, denoting the index of a decision tree and the total number of decision trees in the gradient boosting model; r_{im}, representing the residual of the i^{th} sample in the m^{th} decision tree; R_{jm}, denoting the j^{th} leaf node of the m^{th} decision tree; α, referring to the learning rate when building the model.

Table 6.1 Pseudocode and mathematics of gradient boosting for regression.

Input: Training dataset $\{(x_i, y_i)\}_{i=1}^{n}$ and a differentiable Loss Function $L(y_i, F(x_i)) = \frac{1}{2}[y_i - F(x_i)]^2$
Step 1: Initialize model with a constant value $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$
Step 2: for $m = 1\ to\ M$:
(a) Find pseudo-residuals $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for $i = 1,2, \ldots, n$
(b) Fit the regression tree to the training dataset $\{(x_i, r_{im})\}_{i=1}^{n}$
(c) For $j = 1 \ldots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma)$
(d) Update $F_m(x_i) = F_{m-1}(x_i) + \alpha(\gamma_{jm} \vert x_i \in R_{jm})$
Step 3: Output $F_m(x)$

CHAPTER 7

Support Vector Machine

Abstract: In this chapter, we investigate Support Vector Machines (SVM) for both linearly separable and linearly non-separable cases, emphasizing accessibility by minimizing abstract mathematical theories. We present concrete numerical examples with small datasets and provide a step-by-step walkthrough, illustrating the inner workings of SVM. Additionally, we offer sample codes and comparisons with the SVM model available in the scikit-learn library. Upon completing this chapter, readers will gain a comprehensive understanding of SVM's mechanics, and its connection to the implementation and performance of the algorithm, and be well-prepared to apply it in their practical applications.

Keywords: Support Vector Machine, Linearly Separable, Linearly Non-Separable, Polynomial Kernel, Radial Basis Function Kernel, Numerical Example, Small Dataset, Scikit-Learn

7.1. INTRODUCTION TO SUPPORT VECTOR MACHINE

Support vector machine (abbreviated as SVM) is a powerful and widely applicable machine learning algorithm that has been successfully employed across various domains, such as image and speech recognition, natural language processing, bioinformatics, and finance, to name a few. The goal of the SVM algorithm is to determine, in the space of N dimensions (where N is the number of features), a hyperplane that classifies the data points in a clearly distinguishable manner. It is defined in such a way that the margin distance between data points from different classes is maximized in the N-dimensional space [1, 2]. If the margin distance is maximized, then subsequent new data points (previously unseen) will be classified with greater confidence. For the linearly non-separable dataset, SVM primarily employs a kernel function to map the original data to a high-dimensional Hilbert Space to achieve linear separability and thereby resolve the linear non-separable problem [3]. Besides, it is worth mentioning that support vectors are just the training data points that are closer to the hyperplane. These data points are more relevant and critical to constructing an SVM model, as they help determine the equation of the separating hyperplane [4]. Fig. (**7.1**) illustrates a hyperplane, support vectors, and margin using a dataset with 10 samples from 2 classes, namely, positive $(+)$ and negative $(-)$ classes.

Zhiyuan Wang, Sayed Ameenuddin Irfan, Christopher Teoh & Priyanka Hriday Bhoyar

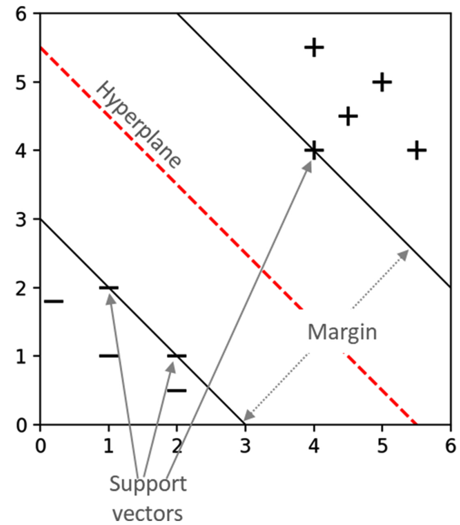

Fig. (7.1). Illustration of SVM hyperplane, support vectors, and margin.

7.2. MATHEMATICS OF SUPPORT VECTOR MACHINE: LINEARLY SEPARABLE CASE

The dataset utilized in Fig. (**7.1**) has been simplified in an effort to reduce complexity. As such, there are now only 2 data points belonging to the positive class ($+$), and 2 data points belonging to the negative class ($-$), as shown in Fig. (**7.2**). The objective is to find the hyperplane that is tied to the maximum margin. Next, we draw a vector $\vec{w}$ (any length) that starts from the origin and is perpendicular to the hypothetical hyperplane. In addition, suppose we also have previously unseen data $\vec{u}$, and we would like to predict the class of $\vec{u}$, whether it is in the $+$ or $-$ class.

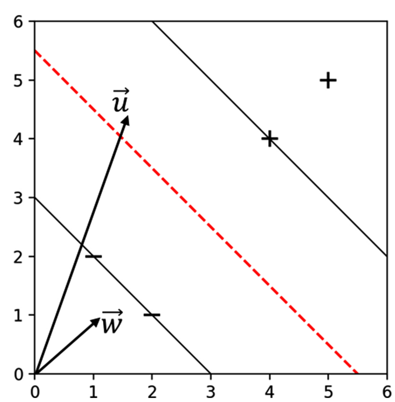

Fig. (7.2). Illustration of 2 data points of $+$ class, 2 data points of $-$ class, $\vec{w}$ perpendicular to the hypothetical margin, and a previously unseen $\vec{u}$.

Project $\vec{u}$ down to the vector perpendicular to the margin *(i.e.,* $\vec{w}$), if that projection is greater than or equal to ($\geq$) certain constant c, which crosses the median line, then it must be a positive ($+$) data point. This can be expressed mathematically as:

$$\vec{w} \cdot \vec{u} \geq c, \text{ then } \vec{u} \text{ is labeled } +$$

Simple transformations are performed:

$$\vec{w} \cdot \vec{u} - c \geq 0$$

$$\vec{w} \cdot \vec{u} + b_t \geq 0, \text{ where } b_t = -\mathrm{c}$$

The median line of the margin:

$$\vec{w} \cdot \vec{u} + b_t = 0$$

The edge line (near $+$) of the margin:

$$\vec{w} \cdot \vec{u} + b_t = \delta$$

Here, δ is a positive constant

Divide both sides by δ:

$$\frac{\vec{w}}{\delta} \cdot \vec{u} + \frac{b_t}{\delta} = 1$$

Let $\vec{W} = \frac{\vec{w}}{\delta}$ and $b = \frac{b_t}{\delta}$,

The edge line (near $+$) of the margin is updated to:

$$\vec{W} \cdot \vec{u} + b = 1$$

The median line of the margin is updated to:

$$\vec{W} \cdot \vec{u} + b = 0$$

Symmetrically, as shown in Fig. (**7.3**), the edge line (near $-$) of the margin is updated to:

$$\vec{W} \cdot \vec{u} + b = -1$$

CHAPTER 8

K-means Clustering

Abstract: In this chapter, we explore the K-means clustering algorithm, emphasizing an accessible approach by minimizing abstract mathematical theories. We present a concrete numerical example with a small dataset to illustrate how clusters can be formed using the K-means clustering algorithm. Additionally, we provide sample codes and comparisons with the K-means model available in the scikit-learn library. Upon completing this chapter, readers will gain a comprehensive understanding of the mechanics behind K-means clustering, and its connection to the implementation and performance of the algorithm, and be well-prepared to apply it in practical use.

Keywords: K-Means Clustering, Distance Metrics, Numerical Example, Small Dataset, Scikit-Learn

8.1. INTRODUCTION TO CLUSTERING AND DISTANCE METRICS

In unsupervised learning, the algorithm is not provided with labeled training data. Instead, it is only given a set of input examples, and the primary objective of unsupervised learning is to uncover the inherent structure or patterns within the data, enabling the algorithm to make predictions, decisions, or recommendations based on these discovered patterns [1]. This contrasts with supervised learning, where the algorithm is given both input examples and corresponding labeled outputs and can learn by making predictions and comparing them to the true labels. The ability of unsupervised learning algorithms to discover hidden structures and relationships in the data without relying on labeled examples makes them particularly valuable in situations where obtaining labeled data is challenging, time-consuming, or expensive.

Unsupervised learning has a variety of applications, such as anomaly detection, clustering, and dimensionality reduction. For example, an unsupervised learning algorithm might be used to cluster customers based on their usage of electronic devices, with the goal of identifying potential users of blue light filter lenses. One cluster may consist of customers who spend a significant amount of time on screens and use multiple devices frequently, indicating that they may be potential users of blue light filter lenses. Another cluster may consist of customers who use electronic devices infrequently, indicating that they may not be interested in purchasing blue light filter lenses. The clustering information can provide valuable insights for marketing efforts and enable precise targeting of potential customers.

Zhiyuan Wang, Sayed Ameenuddin Irfan, Christopher Teoh & Priyanka Hriday Bhoyar

There are numerous unsupervised machine learning algorithms available, including K-means clustering, principal component analysis, and hierarchical clustering. In this chapter, we will delve into the details of K-means clustering. Before using the K-means clustering algorithm, it is important to note that distance metrics are crucial for accurately measuring the distance between data points in two to n-dimensional space and forming appropriate clusters. There are four popular distance metrics, namely, Euclidean distance, Manhattan distance, Cosine similarity, and Chebyshev distance.

8.1.1. Euclidean Distance

Euclidean distance is a commonly used distance metric that calculates the distance between two points by determining the shortest path between them. The formula for calculating Euclidean distance is the square root of the sum of the squared differences in the coordinates of the two points. This measure is useful for understanding the relationship between data points in a multi-dimensional space.

Euclidean distance between points A and B is defined as:

$$D_{AB} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + \cdots + (z_2 - z_1)^2} \quad \textbf{(8.1)}$$

Here, the coordinate of point A is $(x_1, y_1, \dots, z_1)$ and point B is $(x_2, y_2, \dots, z_2)$.

8.1.2. Manhattan Distance

Manhattan distance, also known as the taxicab distance, is a distance metric that calculates the distance between two points by adding up the absolute differences in their coordinates. It is called the taxicab distance because it represents the distance that a taxicab would have to travel to get from one location to another if it could only move horizontally or vertically.

Manhattan distance between points A and B is defined as:

$$D_{AB} = |x_2 - x_1| + |y_2 - y_1| + \cdots + |z_2 - z_1| \quad \textbf{(8.2)}$$

Here, the coordinate of point A is $(x_1, y_1, \dots, z_1)$ and point B is $(x_2, y_2, \dots, z_2)$

8.1.3. Cosine Similarity

The cosine similarity is a distance metric that determines how similar two vectors are to one another by computing the cosine of the angle that separates them.

Cosine similarity between A and B is defined as:

$$\cos(\theta) = \frac{A \cdot B}{\|A\| \, \|B\|} \tag{8.3}$$

Here, θ is the angle between the vectors A and B.

$A \cdot B$ is the dot product of vectors A and B

$\|A\|$ and $\|B\|$ are L2 norm of the vectors A and B, respectively.

8.1.4. Chebyshev Distance

Chebyshev distance, also known as the chessboard distance, is another metric that measures the distance between two vectors in a vector space. It is calculated by determining the greatest difference between the two vectors along any coordinate dimension.

Chebyshev distance between points A and B is defined as:

$$D_{AB} = Max(|x_2 - x_1|, |y_2 - y_1|, \ldots, |z_2 - z_1|) \tag{8.4}$$

Here, the coordinate of point A is $(x_1, y_1, \ldots, z_1)$ and point B is $(x_2, y_2, \ldots, z_2)$.

8.2. ALGORITHM OF K-MEANS CLUSTERING

K-means clustering was first developed by Stuart Lloyd at Bell Labs in 1957 for pulse-code modulation. The idea was not publicly published outside of the company until 1982. The K-means clustering algorithm is also known as the Lloyd-Forgy algorithm due to the development of a nearly identical method by Edward W. Forgy in 1965 [2]. It has since become a widely used algorithm in the field of unsupervised machine learning for clustering data into groups with similar characteristics.

Let us understand the working of the K-means algorithm and how it forms clusters. If we were given the centroids for an unlabeled dataset, it would be easy to label all the samples by assigning each of them to the cluster with the closest centroid. On the other hand, if we were provided with the labels for all the samples, we could easily find all the centroids by calculating the mean of the samples for each cluster. However, since neither the labels nor the centroids are given to us, it is unclear how to proceed. To get started, we can simply place the centroids randomly (*e.g.,* by selecting *k* samples at random and using their positions as the initial centroids). The

SUBJECT INDEX

Zhiyuan Wang, Sayed Ameenuddin Irfan, Christopher Teoh & Priyanka Hriday Bhoyar

O

P

R

S

T

V

W