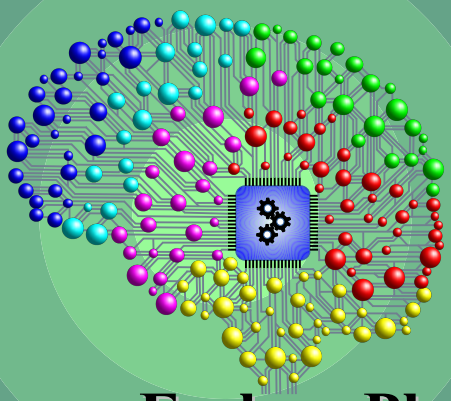
ADVANCED MACHINE LEARNING MADE EASY

FROM THEORY TO PRACTICE WITH NUMPY AND SCIKIT-LEARN

VOLUME 1 Generalized Linear Models



Ferenc Farkas, Ph.D.

Draft:Advanced Machine Learning Vol.1 Author:Ferenc Farkas Date:13-Jan-2022

Ferenc Farkas, Ph.D.

Advanced Machine Learning Made Easy

From Theory to Practice with NumPy and scikit-learn

Volume 1: Generalized Linear Models

Draft: Advanced Machine Learning Vol.1 Author: Ferenc Farkas Date: 13-Jan-2022

This book is for sale at
http://leanpub.com/AML1

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

To the communities creating iPython, Jupyter Notebook, SciPy, NumPy, pandas, matplotlib, statsmodels, and scikit-learn

In memoriam:

Szabó, György Pál
dr. Weszely, Tibor

Preface

Machine learning is a highly interdisciplinary topic and refers to a set of tools for modeling and understanding complex datasets. It is a field of computer science that uses statistical techniques to give computer systems the ability to "learn" with data (e.g., progressively improve their performance on a specific task), without being explicitly programmed. In other words, machine learning gives us the technology that allows us to automatically learn complex rules efficiently from a set of examples in a way that is much more accurate and flexible than attempting to program all the rules by hand. It is the most exciting field of all computer science, having a large impact on many applications. With the explosion of "Big Data" problems, machine learning has become a very hot field in many scientific areas, as well as marketing, finance, and other business disciplines. People with machine learning skills are in high demand, and data scientist positions are advertised everywhere.

This series of books with three volumes cover a wide variety of topics in machine learning, focusing only on supervised and unsupervised learning without discussing the neural networks with deep learning or reinforcement learning. The primary goal of this series is to help readers gain a deep understanding of the concepts, techniques, and mathematical frameworks used by experts in machine learning. This book series is intended for both undergraduate and graduate students, as well as software developers, experimental scientists, engineers, and financial professionals with strong math backgrounds who wish to improve their machine learning skills. This book series is also intended for those who want to become a data scientist, as well as data science practitioners interested in deepening their theoretical knowledge.

There are myriads of articles, blogs, and ebooks available online, so the natural question would be: why another book in machine learning? Although these online resources are great, and many of them were used intensively during writing these books, none of them filled the author's needs: some of them were too superfluous, others were too theoretical, or some were great but focused to only a particular set of machine learning algorithms. Thus, this three-volume book shall be seen as a compilation of the thousands of information chips gathered from different materials and put together to form a concise description of this emerging field.

There are several ways to start learning a new field of interest, the main two approaches are: from top to down and from bottom to up. This three-volume book series takes the latter approach by building up first the basic concepts and then providing the mathematical framework to build up each machine learner step-by-step. The greatest challenge for a newbie in a new field of study is to learn all the new buzzwords. And to make things even harder, some buzzwords could have multiple meanings, while others can be used interchangeably for the same thing. This book series attempts to solve this pitfall by using intensive indexing and by highlighting new buzzwords when introduced the first time in the book.

To make this book an easy read, some mathematical background, equivalent to a one-semester undergraduate course, in each of the following fields is preferred: linear algebra, multivariate differential calculus, probability theory, and statistics. Thus, it is assumed that the reader has familiarity with basic probability and statistics: knows what random variables are, how expectation and variance of a random variable are defined, heard about the common probability distributions and their properties, etc. It is also assumed that the reader is familiar with the notion of vectors and matrices, knows how to multiply vectors and matrices, and understand what the matrix inverse, determinant, and eigenvalues are. The reader should be also familiar with the notion of first and second-order (partial) derivatives, how to calculate them, respective should know the Lagrangian function. Although probability theory and statistics were also listed here, a great deal of emphasis is put in the book on repeating some concepts related to these fields. It is also assumed that the reader does have some sort of basic knowledge of computer science and possess knowledge of the basic computer skills and principles, including, but not limited to, data structures and algorithms. Basic programming skills, some knowledge of Python programming, the SciPy stack, and Jupyter Notebook is also required from the reader to carry out the lab exercises accompanying the book.

This book series can be also seen as a very concise description of the algorithms included in scikit-learn library, but the primary goal is to provide a good understanding of not just the API of scikit-learn (and statsmodels), but also to give a good comprehension about what is under the hood, how they are working and what are the pros and cons of each machine learning algorithm. No single approach will perform well in all possible applications - as no universal machine learning algorithm exists that works well on all problems - thus, without understanding all the cogs and their interaction inside the machine (learner), it is impossible to select the best algorithm for a particular problem.

Although reading the three-volume series requires a solid math background, those who lack the necessary math skill should not run away in panic. The author is an engineer and not a mathematician who sees the math only as a tool that serves as a way to deepen the understanding of a problem at hand and to find the optimal solution for a practical problem. Thus, all mathematical formulas used by machine learning algorithms are introduced by formulating the background and providing additional intuition beforehand. With that in mind, minimal mathematical knowledge might be also acceptable for an eager learner to understand the book. Moreover, the mathematical expression for each machine learning algorithm is derived step by step

with clear explanations, intuitive examples, and supporting figures. For those not possessing a deep mathematical background - but have some programming knowledge - should see the vectors and matrices used in the mathematical framework as the counterpart of multidimensional arrays used in computer programs, while the mathematical formulas as a sequence of array manipulations. Thus, the mathematical formulas presented in the book will be converted directly into a single line of Python code using array manipulations. This approach is supported by both the Appendix and lab exercises. As such, the theoretical knowledge can be deepened with the lab exercises stored in GitHub as Jupyter notebooks.

Lastly, for those who have instinct reluctance toward mathematical expressions, I would recommend the scene from the famous Matrix movie where the actors were able to decipher instantly what in the Matrix simulation was going on just by watching the falling strange green characters on the screen. And this is not fiction, but rather science: our brain is very plastic even at old age according to Neuroplasticity and is capable of rewiring itself for carrying out whatever task is needed to be performed. So, even if you are thinking that you are not capable of understanding mathematics, this is more your unconscious belief than your real capability. Just be patient with yourself and remember: your brain needs time to rewire itself to comprehend the mathematical symbols, just like it needs time to learn all the traffic signs to be able to drive. And when you achieve this with practice, you will read and comprehend the mathematical expressions just like any ordinary text.

Budapest, January 2020 *Farkas, Ferenc*

Acknowledgments

Firstly, I want to thank all the communities for creating such great tools as the open-source Python programming environment with all its wonderful libraries including but not limited to, SciPy, NumPy, pandas, matplotlib, seaborn, scikit-learn, and statsmodels. I also want to especially thank the communities creating the free cross-platform MiKTeX and TeXstudio without which this book would not exist. All pictures shown in the book were created with LibreOffice Draw and, especially, matplotlib library (libraries using matplotlib are also included).

Secondly, I want to thank all the universities that made possible the online access of their up to date courses in the field of Machine Learning, Statistical Inference, Data Science, and Data Mining through different Massive Open Online Course platforms or on their websites. I also owe a big thank to the authors of blogs and to professors associated at different universities around the world making publicly available lecture notes.

Thirdly, I want to thank my family: to my wife and my son, for being patient and supporting me in writing this book. There is a great saying: "Behind every successful man there is a strong woman", and undoubtedly this is true in my case, as well. The background support received from my wife was necessary for writing the three-volume series of books.

Last, but not least, I owe a deep sense of gratitude for those who provide me feedback and help me shape this book series.

Contents

Acronyms

ACC	Accuracy
ACF	AutoCorrelation Function
AIC	Akaike Information Criterion
AI	Artificial Intelligence
API	Application Programmable Interface
AUC	Area Under the Curve
BIC	Bayesian Information Criterion
BLUE	Best Linear Unbiased Estimator
CD	Coordinate Descent
CDF	Cumulative Distribution Function
CG	Conjugate Gradients
CI	Confidence Interval
CLT	Central Limit Theorem
CNN	Convolutional Neural Networks
CV	Cross Validation
EDA	Exploratory Data Analysis
ERM	Empirical Risk Minimization
ERR	Error Rate
FGLS	Feasible Generalized Least Squares
FN	False Negative
FNR	False Negative Rate
FP	False Positive
FPR	False Positive Rate
GD	Gradient Descend
GLS	Generalized Least Squares
GV	Generalized Variance
IG	Information Gain
IID	Independent and Identically Distributed
INT	Inverse Normal Transformation
IQR	InterQuartile Range
LCL	Lower Confidence Level

MAE Mean Absolute Error
MadAE Median Absolute Error
MAP Maximum a Posteriori Estimation
MAR Missing At Random
MCAR Missing Completely At Random
MCC Matthew's Correlation Coefficient
MCD Minimum Covariance Determinant
MI Mutual Information
MLE Maximum Likelihood Estimator
MLR Multiple Linear Regression
MSB Mean Squares Between-groups
MSE Mean Squared Error
MSR Mean Squared Residual
MSW Mean Squares Within-groups
NLP Natural Language Processing
NMAR Not Missing At random
NPV Negative Predicted Value
OCR Optical Character Recognition
OLS Ordinary Least Squares
PAC Probably Approximately Correct
PACF Partial AutoCorrelation Function
PC Principal Component
PCA Principal Components Analysis
PCR Principal Components Regression
PDF Probability Density Function
PMF Probability Mass Function
PPF Percent-Point Function
PPV Positive Predictive Value
PRE Proportional Reduction in Error
RMSE Root Mean Square Error
ROC Receiver Operating Characteristics
RSE Residual/Regression Standard Error
RSS Residual Sum of Squares
SAG Stochastic Average Gradient
SF Survival Function
SGD Stochastic Gradient Descent
SLR Simple Linear Regression
SRM Structural Risk Minimization
SSB Sum of Squares Between-groups
SSE Error Sum of Squares
SSR Regression Sum of Squares
SST Total Sum of Squares
SSW Sum of Squares Within-groups
SVD Singular Value Decomposition
TN True Negative

TNR	True Negative Rate
TP	True Positive
TPR	True Positive Rate
TSA	Time Series Analysis
UCL	Upper Confidence Level
VIF	Variance Inflation Factor
WLS	Weighted Least Squares
WSSE	Weighted Error Sum of Squares
WSSE	Weighted Total Sum of Squares
ZCA	Zero-phase Components Analysis

Symbols

$\neg$	logical NOT
x	scalar (also denotes the observed value of a random variable)
$\|x\|$	absolute value of a scalar
$\|S\|$	cardinality of a set
$x \cdot y$	scalar product (only explicitly used to separate variable names)
x^p	power to p of a scalar value (p represents a real number)
$\sqrt{x}$	square root of the scalar value (same as $x^{\frac{1}{2}}$ or $x^{0.5}$)
$\mathcal{X}$	input space or instance space
$\mathcal{Y}$	output space or label space
D	number of input features (dimension of the input space)
M	number of outputs (dimension of the output space)
K	number of classes (number of unique labels in a classification problem)
N	number of observations in a sample (number of entries in a dataset)
$\mathcal{D}_{N,D}$	dataset with N entries (observations) and D features (input variables)
$N \times D$	dimension of a matrix (N number of rows, D number of columns)
$i = \overline{1,N}$	i takes the values of $1, 2, 3, ...,$ till N
$\mathbf{x}$	vector (always represents a column vector)
$\mathbf{x}^\top$	transpose of the vector (i.e. a row vector)
$\mathbf{X}$	matrix (also input matrix created from dataset with no bias included)
$\dot{\mathbf{X}}$	design matrix (input matrix with bias - first column of ones - included)
$\mathbf{X}^\top$	transpose of a matrix
$\mathbf{X}^{-1}$	inverse of a matrix
$\mathbf{X}^p$	element-wise power to p of the matrix (the same applies to vectors)
$\|\mathbf{X}\|$	determinant of the matrix
$[\mathbf{X}]_{ij}$	element x_{ij} of the matrix
$\text{tr}(\mathbf{X})$	trace of the matrix (sum of diagonal elements of a square matrix)
$\text{rank}(\mathbf{X})$	rank of the matrix (max. number of linearly independent columns)
$\textbf{diag}(\mathbf{X})$	main diagonal elements of a square matrix (the result is a vector)
$\textbf{Diag}(\mathbf{x})$	diagonal matrix created from a vector
$\mathbf{I}$	identity matrix
$\mathbf{X} + \mathbf{Y}$	matrix element-wise addition (the same applies to vectors)

$\mathbf{X}-\mathbf{Y}$	matrix element-wise subtraction (the same applies to vectors)
$\mathbf{X}\circ\mathbf{Y}$	matrix element-wise multiplication (the same applies to vectors)
$\frac{\mathbf{X}}{\mathbf{Y}};\mathbf{X}/\mathbf{Y}$	matrix element-wise division (the same applies to vectors)
$\mathbf{X}\cdot\mathbf{Y}$	matrix dot product (only explicitly used to separate variable names)
$\mathbf{X}\cdot\mathbf{y}$	dot product between matrix and vector
$\mathbf{X}\cdot y$	matrix element-wise product with a scalar (the same applies to vectors)
$\mathbf{x}^\top\cdot\mathbf{y}$	vector inner product (the result will be a scalar)
$\mathbf{x}\cdot\mathbf{y}^\top$	vector outer product (the result will be a matrix)
$\sum_{i=1}^{N} x_i$	sum of the elements x_i for $i=\overline{1,N}$
$\prod_{i=1}^{N} x_i$	product of the elements x_i for $i=\overline{1,N}$
$\Sigma\mathbf{x}$	sum of the elements of the vector (the result is a scalar)
$\Sigma\mathbf{X}$	sum of the elements of the matrix (the result is a scalar)
$\Sigma_r\mathbf{X}$	sum of elements of each row of the matrix (the result is a vector)
$\Sigma_c\mathbf{X}$	sum of elements of each column of the matrix (the result is a row vector)
x_j	the j-th element of a vector (the j-th feature of a feature vector)
$x^{(i)}$	the i-th observation as a scalar (when there is only one feature)
$(\mathbf{x}^{(\mathbf{i})})^\top$	the i-th observation as a vector (when there are multiple features)
$\mathbf{x}_j$	all observations of the j-th feature (a vector with N observations)
$x_j^{(i)}$	the j-th feature from the i-th observation
ε	random error
X	random variable
$\overline{x}$	average value of observations of the random variable X (sample mean)
$\widetilde{x}$	median value of observations of the random variable X (sample median)
μ	population mean
σ	population standard deviation
σ^2	population variance
$p(X)$	probability density function of the continuous random variable X
$P(X)$	probability mass function of the discrete random variable X
$F(X)$	cumulative distribution function of the random variable X
$\mathbb{P}[X]$	probability of the random variable X
$\mathbb{P}[X\|Y]$	conditional probability of the random variable X given Y
$\mathbb{E}[X]$	expected value of the random variable X
$\mathbb{V}[X]$	variance of the random variable X
$\mathbb{S}[X]$	skewness of distribution of the random variable X
$\mathbb{K}[X]$	kurtosis of distribution of the random variable X
$\mathbb{C}[X,Y]$	covariance between random variable X and Y
$\boldsymbol{\Sigma}$	covariance matrix of random variables
S	sample standard deviation
S^2	sample variance
S_{xy}	sample covariance between random variables X and Y
$\mathbf{S}$	sample covariance matrix
$\hat{\theta}$	estimated value (estimator of θ)
$\mathbb{B}[\hat{\theta}]$	bias of an estimator $\hat{\theta}$
$f(x)$	function of scalar x (univariate function)
$f(\mathbf{x})$	function of vector $\mathbf{x}$ (multivariate function of the form $f(x_1,x_2,\cdots,x_D)$)

$\mathbf{f}(\mathbf{x})$	vector function of vector $\mathbf{x}$ of the form $f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_M(\mathbf{x})$
$\frac{\mathrm{d}f}{\mathrm{d}x}$	derivative of the function $f(x)$ with respect to variable x
$\frac{\partial f}{\partial x_j}$	partial derivative of the function $f(\mathbf{x})$ with respect to the j-th element of the vector $\mathbf{x}$
$\nabla_{f(\mathbf{x})}$	gradient of the function f (vector of partial derivatives with elements $\frac{\partial f}{\partial x_j}$ for $j = \overline{1,D}$)
$\mathbf{J}$	$M \times D$ Jacobian matrix of a vector function $\mathbf{f}(\mathbf{x})$ with elements $\mathbf{J}_{ij} = \frac{\partial f_i}{\partial x_j}$
$\mathbf{w}$	vector of model parameters (also called weights)
$J(\mathbf{w})$	cost function or loss function
$L(\mathbf{w})$	likelihood function
ℓ	logarithm of the likelihood function (log-likelihood for short)
$\mathcal{L}(\mathbf{w})$	Lagrange function (or Lagrangian)
$\mathfrak{R}(.)$	rank of an ordinal variable
$\mathcal{R}(.)$	(statistical) risk (generalization error)
exp	natural exponent (applied element-wise to vectors/matrices)
log	natural logarithm (applied element-wise to vectors/matrices)
$\log_b$	logarithm with base b (applied element-wise to vectors/matrices)
$\propto$	proportional to
$\cong$	approximately equal
$\overset{set}{=}$	set to be equal to
$\equiv$	equivalent (due to definition or notation)
$\gg$	much greater than
$\ll$	much less than
B	backshift operator
Δ	difference operator
$\|\mathbf{x}\|_p$	p-norm (a function assigning a positive length to a vector)
$I\{.\}$	indicator function (has value of 1 for a true attribute, 0 otherwise)
$\min\{\mathbf{x}\}$	minimum value of vector $\mathbf{x}$ (the same applies to matrices)
$\inf\{.\}$	infimum value of a set
$\max\{\mathbf{x}\}$	maximum value of vector $\mathbf{x}$ (the same applies to matrices)
$\sup\{.\}$	supremum value of a set
$\arg\min\{\mathbf{x}\}$	the index of the minimum value of vector $\mathbf{x}$ (the same applies to matrices)
$\arg\max\{\mathbf{x}\}$	the index of the maximum value of vector $\mathbf{x}$ (the same applies to matrices)
$\min_x\{.\}$	provides the minimum value of the attribute for a given x
$\max_x\{.\}$	provides the maximum value of the attribute for a given x
$\arg\min_x\{.\}$	provides the value of x for which the attribute has minimum value
$\arg\max_x\{.\}$	provides the value of x for which the attribute has maximum value
$\lim_{x\to\infty}\frac{1}{x}$	the limit of $\frac{1}{x}$ as x approaches to infinity
$\mathcal{O}(.)$	computational cost

Chapter 1
Introduction

1.1 How to Use the Book

The intent of this book, as the subtitle of the book series suggests, is to start from the theory but get comprehension with practice - learning by example is the most efficient way of learning. In the book, mathematical equations for deriving different machine learning algorithms are extensively used. This helps in deepening our understanding of how each algorithm works, what are their limitations, respective what are the benefits or drawbacks of using each of them. Tables and figures are also inserted in the text to show the results of each algorithm for real-world problems and also highlighting their advantages and disadvantages.

`Names` of data sets and their variables - used as examples in the book - are emphasized with a typewriter font, while ***new concepts*** - also listed in the Index at the end of the book - are highlighted with a bold italic font. Emphasizing **domains** of applications is done by bold font.

Important explanations, statements or remarks are placed in a grayed box to call attention to.

All references to sections, figures, tables, equations, and external resources are provided as hyperlinks, thus, while reading the Ebook one can quickly jump to the appropriate reference whether it is internal or external to the book.

Deriving machine learning algorithms are done step by step starting with the problem formulation. Vectorization is also introduced which helps to omit, or at least minimize, for loops. The final equations(s) of an algorithm can be easily transformed into a single line of Python/NumPy code.

Note *Explaining mathematical concepts using intuition or adding important notes to new concepts are highlighted with italic font and a shaded box containing "Note" is placed in the front.*

Algorithms with pseudo-code are placed in a framed box with the caption as shown below:

Algorithm 1.1 Example of a factorial algorithm

```
1: function FACTORIAL(x)
2:     y=1
3:     while x>0 do
4:         y=y*x
5:         x=x-1
6:     end while
7:     return y
8: end function
```

Because of the rapid development of the open-source tools, the book does not include any code (except for a few in the Appendix B) but focuses mainly on the theory bundled with practical examples. Thus, at the end of each chapter, several lab exercises are listed for practicing. However, the book per se does not contain solutions to the exercises. Instead, complete solutions with explanations can be obtained from the accompanying Jupyter Notebooks[1], that can be found at https://github.com/FerencFarkasPhD/Advanced-Machine-Learning.

The accompanying Jupyter Notebooks provides supplementary material to the book and should be read (and code executed) in parallel with reading this book. The implemented algorithms have the sole purpose of deepening the learner's comprehension and they are not targeted for any optimization. For big dataset[2] the well-optimized scikit-learn library shall be used instead as shown at the end of the accompanying Jupyter notebooks.

1.2 Definition of Machine Learning

There are dozens of terms circulating nowadays around ***machine learning*** discipline, so let's, make some clarification around them. First, we should start with ***artificial intelligence*** (AI), sometimes called machine intelligence, which is intelligence demonstrated by machines, in contrast to the natural intelligence displayed by humans and other animals.

AI represents a broader field including knowledge-base databases or other explicitly programmed algorithms for a given task which provides some "intelligence" for the machine. In brief, AI tries to make computers intelligent to mimic the cognitive functions of humans. Machine learning is part of AI but meant for algorithms that provide the ability for the machines to learn by themselves from their environment.

[1] Jupyter Notebook has the main advantage that, besides the code that can be executed inside the notebook, can contain the result of code execution including graphs and plots, thorough explanations for each part of the code, hyperlinks, images, videos, and mathematical formulas.

[2] Written as 'dataset' instead of 'data set' throughout the book as defined by https://dictionary.cambridge.org/dictionary/english/dataset.

It teaches the computer to solve problems by looking at hundreds or thousands or even millions of examples, learning from them, and then using that experience to solve the same problem in new situations. This idea is captured by Arthur Samual, an American pioneer in the field of computer gaming and artificial intelligence, who also coined the term "machine learning" in 1959 while at IBM:

> "Machine learning is the subfield of computer science that gives computers the ability to learn without being explicitly programmed." [1]

To grasp this idea better, let's take an example of classifying cats vs. dog images. In a traditional approach you would try to create a program with rules, like: does the image shows the eyes, and if yes, what are their size and colors?; does the animal has ears, and what is their size, form and color?; and so on. You already can figure out how difficult would be to create such a rule set which is not dependent on a particular dataset. On the contrary, a machine learning algorithm would learn the patterns of each animal on its own while exposing to it thousands or even millions of images of cats and dogs. So, machine learning algorithms, inspired by the human learning process, iteratively learn from data and allow computers to find hidden insights.

Dating the beginning of any movement is difficult, but the Dartmouth Summer Research Project of 1956 is often taken as the event that initiated AI as a research discipline [12], and in the years since has experienced several waves of optimism, followed by disappointment and the loss of funding (known as an "AI winter"), then followed by new approaches with success. Since 2012 another hype emerged in the field of AI, especially for a subfield of machine learning, namely ***deep learning***. Deep learning is more a marketing term as the main concept is the ***neural networks*** which already appeared in the 1950s and it was inspired by the human brain. To emphasize that these neural networks have many hidden layers, the term deep neural network was coined. Neural networks, like any other machine learning algorithms, require training. The training process for deep neural networks is called deep learning. According to Yoshua Bengio from Montreal University the

> "Machine learning research is part of research on artificial intelligence, seeking to provide knowledge to computers through data, observations and interacting with the world. That acquired knowledge allows computers to correctly generalize to new settings." [8]

Several definitions were already provided for machine learning, but a more formal definition was given by Tom Mitchell in 1998:

> "A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E." [14]

To comprehend better the above definition, let's take the spam detection example. Suppose your email program watches which emails you do or do not mark as spam, and based on that the computer learns how to better filter spam. The detection of

spam represents the task T, the computer watching how you classify a given email is the experience E, and the fraction of emails classified correctly as the performance measure P.

In Fig. 1.1 it is shown how the AI, machine learning and neural network with deep learning is interrelated with each other. In this book series, only the machine learning algorithms are discussed without taking a tour in the subfield of neural networks. The latter is so vast that it needs to be covered in a separate book.

In order to have a better idea of the capabilities of machine learning, let's look at some real-world examples. The following list is far from being complete and in the book, several examples are provided for each application mentioned below:

- **Email Spam Detection**: Given Emails in an inbox, identify those Email messages that are spam and those that are not. A machine learning algorithm would allow for the Email application to leave non-spam Emails in the inbox and move spam Emails to a spam folder.
- **Handwritten Digit Recognition**: Given handwritten zip codes on thousands of envelopes in a post office identify the digit for each handwritten character. A computer program with machine learning algorithm would be able to read and understand handwritten zip codes and sort envelops by geographical regions.
- **Credit Card Fraud Detection**: Given credit card transactions for a customer from the past identify those which are genuine (card transaction initiated by the customer) and those that are fraudulent (someone misusing the customer's card). A machine learning algorithm could check in real-time each transaction, and those identified as fraudulent would not be served.
- **Product Recommendation**: Given a purchase history for a customer and a large inventory of products, identify those products of which that customer would be interested in and likely to purchase. Then a machine learning algorithm would make recommendations to the customer for other similar products and likely to motivate for further product purchases.
- **Medical Diagnosis**: Given the symptoms exhibited in a patient and a database of anonymous patient records, predict whether the patient is likely to have an

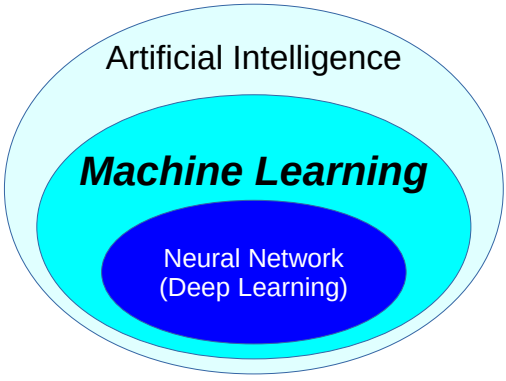


Fig. 1.1 Relation of machine learning with artificial intelligence and neural network with deep learning

illness or not. A machine learning model of this decision problem could be used to provide decision support to medical professionals.

- **Housing Price Prediction**: Given a database with houses sold in the past make predictions on the price of new houses based on their size, number of rooms, geographical location, closeness to public transportation, etc.
- **Social Media Analytics**: Seeking relevance between the brand/product being monitored and millions of blogs talking about this product or competitor's similar product in order to find out how well the monitored product is received in the market compared to the competitors. Identifying a subset of relevant blogs with authoritative bloggers or influential bloggers who are very well connected, and are most responsible for the spread of information in the blogosphere. Then, given the constant chatter on the blogosphere, much insight can be gleaned by examining patterns of what people are blogging about to find emerging areas of discussion.
- **Web search**: Find relevant searches, predict which results are most relevant to the customer and return a ranked output. The ranking is based on what the customer is most likely to click on.
- **Document Classification**: Machine learning based computational approaches largely solves the tedious task of classification of huge amounts of documents based on their types. For example, in a library the books can be classified by a machine learning algorithms as fiction, literature, scientific etc. without requiring a person to go through the content of all the books.
- **Sentiment Analysis**: This is part of ***natural language processing*** (NLP) concerned with the interactions between computers and human (natural) languages, in particular how to program computers to process and analyze large amounts of natural language data. One example of sentiment analysis is to analyze the text reviews of customers for a product or a service and find out whether a given review is more positive or more negative (or even classify the reviews in five categories like giving marks from 1 to 5). Then a machine learning algorithm could automatically provide information about how many positive and negative reviews have been provided for a given product or service.
- **Information Extraction**: It is the process of extracting information from web pages, articles, blogs, business reports, and e-mails. The output is in a summarized form such as an Excel sheet or a table in a relational database. In this respect the above three items (web search, document classification, and sentiment analysis) can be seen as examples of information retrieval. But other form of information extraction are also possible.

1.3 Data Science and Machine Learning

Data science is the art of uncovering the insights and trends that are hiding behind data. It involves data and science, and machine learning is part of that science. In fact, data science is more about science and less about data. A data scientist is a person who attempts to work with data to find answers to questions that (s)he is exploring. If you have curiosity and if you have data that you are manipulating

and exploring it, then you are a data scientist. The very exercise of going through analyzing data, trying to get some answers from it, is data science.

The popularity of interest in the area of data science and machine learning comes from the network and data driven society we find ourselves living in. Data science is relevant today because we have tons of data available and dozens of machine learning algorithms. We used to worry about lack of data in the past; now we have a data flood. The software was expensive in the past; now it's open source and free. The computer was slow with limited storage capabilities in the past: now for a fraction of the cost (or even for free), we can have access to cloud resources.

The techniques and methodologies of data science stem from the fields of computer science and ***statistics***[3]. The term data science has appeared in various contexts over the past forty years but did not become an established term until recently. In 1974, Peter Naur published the "Concise Survey of Computer Methods", which freely used the term "data science" in its survey of the contemporary data processing methods that are used in a wide range of applications.

In November 1997, C.F. Jeff Wu gave the inaugural lecture entitled "Statistics = Data Science?" [23] for his appointment to the H. C. Carver Professorship at the University of Michigan. In this lecture, he characterized statistical work as a trilogy of data collection, data modeling/analysis, and decision making. In his conclusion, he initiated the modern, non-computer science, usage of the term "data science" and advocated that statistics be renamed data science and statisticians data scientists.

However, data science is more than statistics. ***Descriptive statistics*** is solely concerned with properties of the observed data, and it does not rest on the assumption that the data come from a larger population. ***Statistical inference*** is the process of using data analysis (the process of inspecting, cleansing, transforming, and modeling data) to deduce properties of an underlying probability distribution. But Data Science involves some enlargement of academic statistics and machine learning. According to David Donoho [6] data science involves the following activities:

1. **Data Exploration and Preparation**: Data scientist devotes serious time and effort exploring data to sanity-check its most basic properties, to expose unexpected features, and to identify and address anomalies and artifacts in the data. Data cleaning involves reformatting and recoding the data values, and even preprocessing, such as grouping, smoothing, and subsetting.
2. **Data Representation and Transformation**: Data Scientists develop skills in both modern databases (know-how of structures, transformations, and algorithms involved using different representations of data, like text files, spreadsheets, SQL/noSQL databases, distributed databases, and live data streams) and in the mathematical representations of the data (as we will see in the following chapters all real-life data need to be transformed into numbers grouped into arrays or matrices).

[3] The term "statistics" was coined at the beginning of modern era as an effort to compile census data and represents a branch of mathematics dealing with data collection, organization, analysis, interpretation and presentation.

3. **Computing with Data**: Every data scientist should know and use several programming languages for data analysis and data processing, like the popular R and Python, but also specific languages for transforming and manipulating text, and for managing complex computational pipelines. Knowing when to use the right tool for the job is an important attribute. Beyond that, knowledge of cluster and cloud computing and the ability to run massive numbers of jobs on such clusters has become a very important skill to be owned by a data scientist.
4. **Data Visualization and Presentation**: Data scientists often spend a great deal of time making meaningful graphics (they might even develop new form of plots which codifies their understanding of the data), and making good presentations and reports for their customers or stakeholders.
5. **Data Modeling**: Each data scientist in practice uses tools to make inferences and predictions about the data at hand. There are two modeling cultures: ***predictive modeling***, in which one constructs methods which predict well over some specific concrete dataset; ***generative modeling***, in which one proposes a stochastic model that could have generated the data, and derives methods to infer properties of the underlying generative mechanism.
6. **Science about Data Science**: Data scientists are doing science about data science when they identify commonly-occurring analysis/processing workflows and invent new tools and paradigms of computing to change the way data science works.

Before moving on, it might be useful to clarify some new terms, as well. ***Statistical learning*** refers to a set of tools for modeling and understanding complex data sets. It is a recently developed area in statistics and blends with parallel developments in computer science and, in particular, machine learning.

Big data is a term used to refer to the study and applications of data sets that are so big and complex that are beyond the ability of commonly used software tools to capture, curate, manage, and process data within a tolerable elapsed time. Big data makes it possible for you to gain more complete answers because you have more information so you can have more confidence in the predictions. Machine learning can be used as a tool set in analyzing big data.

Data mining is another phrase circulating machine learning and data science and describes the process of discovering patterns in large datasets involving methods at the intersection of machine learning, statistics, and database systems. Data mining is an interdisciplinary sub-field of computer science with an overall goal to extract information (with intelligent method) from a dataset and transform the information into a comprehensible structure for further use (like applying a machine learning algorithm). The term "data mining" is in fact a misnomer, because the goal is the extraction of patterns and knowledge from large amounts of data, not the extraction (mining) of data itself.

A data scientist need to have the following three fundamental skills [15]:

- **Domain Expertise**: Expertise in the domain (where the data is generated) is a must in order to be able to put the right questions and be able to interpret the

results. That is, the more specific your questions are, the more likely you are to get actionable results.

- **Mathematics**: Math is needed in data science like statistical modeling, signal processing, probability models, pattern recognition, and predictive analytics, just to name a few. Data science becomes magical when brilliant mathematical constructs, like machine learning algorithms, are applied to big datasets.
- **Computer Science**: Data science happens inside computer systems and one must know the intricacies of architecture of the physical work environment. Big data requires special storage, special handling and special network capabilities. One should have a good understanding of the required tools and hardware resources when aims to solve a machine learning problem.

In Fig. 1.2 the data science Venn diagram is shown as presented by Palmer S. [15]. As it can be seen machine learning is the frontier between mathematics and computer science and is a great tool for the data scientist. There are other variants of this Venn diagram, like the one of Drew Conway [5]. The only difference is that the Computer Science is replaced with "Hacking skills" and the intersection of it with Domain Expertise (or Substantive Expertise, as he calls it) is called "Danger Zone". Taking into account that data processing is vital in the outcome of a data modeling, danger zone might be an appropriate name for this.

As a conclusion, a good data scientist brings skepticism, experimentation, simulation, and replication to bear on understanding a given phenomena. Moreover, a good data scientist is one who can communicate their findings clearly to others (are good story tellers) and having some fundamental communication skills using tool kits in the form of charting, graphing, and related visualization techniques.

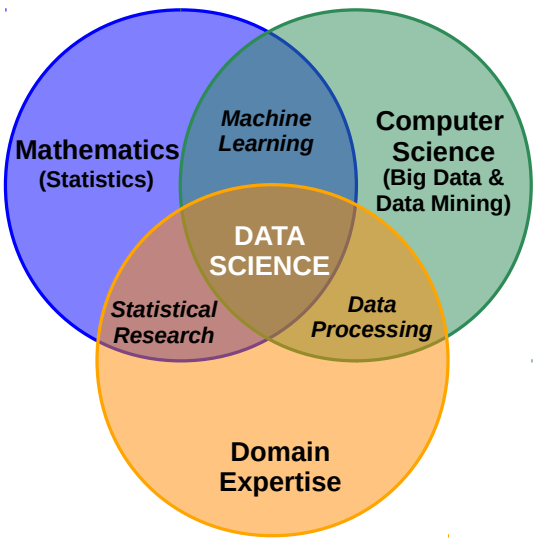


Fig. 1.2 The Data Science Venn Diagram

With this short introduction one should have a better understanding about how these different terms around machine learning are interrelated.

1.4 Types of Machine Learning Algorithm

Machine learning, as its name suggests, learns from available data. However, data can be presented in many forms. Based on how the information is organized we can distinguish between:

- ***Structured data***: Such data is both highly-organized and clearly defined, as well as, easy to digest making analytics possible through the use of legacy data mining solutions. This could be data in a relational database, a spreadsheet, a text file with a tabular structure, and so on. As an example. a structured data is made up largely of basic customer data, which includes names, addresses, and contact information, etc.
- ***Unstructured data***: Data that has some internal structure but is not structured via pre-defined data models or schema. New data sources are made up largely of streaming data coming from social media platforms, mobile applications, location services, and Internet of Things technologies which represent unstructured data. Text files, mobile text messages, Emails, websites, chat, media all represent unstructured data. Since the diversity among unstructured data sources is so prevalent, businesses have much more trouble managing it than they do with old-school structured data.

Data to be used for training a machine learning algorithm, whether is structured or unstructured, can be divided into another two subcategories:

- ***Labeled data***: Data consisting of a set of training examples, where each example is a pair consisting of an input and the desired output value (the latter also called a label). Such data represent data available from the past where humans already defined desirable labels for every example in the dataset. For example, in the case of Email spam detection, every Email in the training data set will have a label associated with it (whether the Email is a spam or not spam).
- ***Unlabeled data***: Data consisting of a set of training examples, where each example represents only the input without providing an associated output. As an example, let's take a bunch of text documents that need to be categorized based on their content (e.g. business, sport, science or technology related) without providing labeled documents. In this case, the machine learning algorithm will categorize the text documents based on their content.

Machine learning algorithms can be divided into four main subcategories based on their learning method:

- ***Supervised learning***: The machine learning model will use labeled training data to learn a link between the inputs and the corresponding outputs. It is called supervised learning because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know

the correct answers, the algorithm iteratively makes predictions on the training data and is corrected by the teacher (the output of the training example). Learning stops when the algorithm achieves an acceptable level of performance. Thus, a supervised learning goal is to determine the function so well between the inputs and the corresponding outputs that when a new input is given, it can predict the output. These types of machine learning models are also called predictive models.[4]

- ***Unsupervised learning***: This type of learning is applied for unlabeled data. The goal of unsupervised learning is to model the underlying structure or distribution in the data to learn more about the data. These are called unsupervised learning because, unlike supervised learning, there are no correct answers (no output in the dataset), thus there is no teacher. Algorithms are left to discover and present by themselves the interesting structure in the data. However, that does not mean that there is no need for human interaction in fine-tuning the unsupervised machine learning model, as we will see in the last volume of the book series. These type of machine learning algorithms are also called descriptive models as they are aimed at summarizing or representing the data structure in a compact manner without providing an underlying causal theory between the variables. [21]
- ***Semi-supervised learning***: Problems where the dataset consists of a large amount of unlabeled data and only a minor part is labeled the so-called semi-supervised learning problems is used. These problems sit in between both supervised and unsupervised learning. Many real-world machine learning problems fall into this area. The main reason is that labeling data by a human is very time-consuming and also expensive as it requires domain expertise. Whereas unlabeled data is cheap and easy to collect and store.
- ***Reinforcement learning***: In reinforcement learning, the goal is to develop a system that improves its performance based on interactions with its environment. Besides the information about the current state of the environment, the so-called reward signal is also provided as feedback to the system. Thus, one can think of reinforcement learning as a field related to supervised learning. However, in reinforcement learning, this feedback is not the correct ground truth label, but a measure of how well the action was measured by a reward function. Through its interaction with the environment, the system can then use reinforcement learning to learn a series of actions that maximize this reward via an exploratory trial-and-error approach. A system using reinforcement learning is somehow similar to a biological organism which learns from the interaction with its surrounding environment.

In this book series, only the first two learning methods will be discussed with a brief discussion about the third item, as well. However, reinforcement learning will not be covered in this book series as it is not part of scikit-learn library.

[4] According to Shmueli [21] in explanatory modeling the function represents an underlying causal function, and the input is assumed to cause the output, while in predictive modeling the function captures the association between input and output. Throughout this book no such distinction will be made.

Supervised and unsupervised learning can be further divided into subcategories. Supervised learning can be categorized based on the type of output in the dataset:

- ***Regression***: When the output is continuous in the labeled dataset and it can take any real value within an interval we talk about regression. In regression, the goal is to find the relationship between the input and its corresponding output. More specifically, regression analysis helps one understand how the typical value of the output changes when any one of the input is varied, while the other inputs are held fixed. Regression analysis is widely used for prediction and forecasting. House price prediction is one example of regression.
- ***Classification***: When the output is discrete in the labeled dataset and can take only a finite number of values (list of categories) we talk about classification. Classification is the problem of identifying to which category a new observation belongs to, on the basis of a so-called training set containing observations whose category membership is known. Examples for classification are: assigning a given Email to the "spam" or "non-spam" class, or assigning a diagnosis to a given patient based on observed characteristics of the patient (gender, blood pressure, age, presence or absence of certain symptoms, etc.).

Unsupervised learning can be also further categorized into several subsets based on the type of problem they are aiming to solve:

- ***Anomaly detection***: This algorithm is used to discover abnormal and unusual cases compared to the dataset already seen. For example, it is used for credit card fraud detection or to observe unusual login attempts to a server.
- ***Dimensionality reduction***: As the number of dimensions increases (which is equal to the number of input variables) the data will become more sparse, i.e. there is no enough data for the learning process. Thus, there is a need to decrease the number of input dimensions so the available data should be enough to create a model which generalizes well.
- ***Clustering***: There is no label associated with each input, thus this type of machine learning algorithm uses a method to assign a set of observations into subsets similar to classification but in an unsupervised fashion. These subsets are known as clusters. The observations inside these clusters are similar to one another, based on some parameters or features. Hence, all the data is divided into clusters without any prior knowledge about the data.
- ***Association*** or ***recommendation systems***: Association rule learning is a rule-based machine learning method for discovering interesting relations between variables in large databases. It is intended to identify strong rules discovered in databases using some measures of interestingness. Its concepts are applied to e-commerce websites, where machines are able to suggest (recommend) other items for the customer to buy, based on the prior purchases they have been made.

1.5 Data Science Methodology

It is important to consider data science methodology, a system of methods used in a data science activity, in your data science work because often there is a temptation

to bypass methodology and jump directly to solutions. However, this strategy might hinder our best intentions in trying to solve a problem. The data science methodology discussed in this section has been outlined by John Rollins from IBM [19]. The Team Data Science Process from Microsoft Azure was also considered in creating this section [7].

In a nutshell, the data science methodology described by John Rollins aims to answer ten basic questions in a prescribed sequence [3].

1. "What is the problem that you are trying to solve?"
2. "How can you use data to answer the question?"
3. "What data do you need to answer the question?"
4. "Where is the current data coming from or how will you get it?"
5. "Is the data that you have a representative of the problem to be solved?"
6. "What additional work is required to work with your data?"
7. "In what way can you present your data to answer the initial question?"
8. "Does your used model really answers the initial question?"
9. "Can you put your model into practice?"
10. "Can you get constructive feedback to adjust your model?"

Like traditional scientists, data scientists need a foundational methodology that will serve as a guide for solving their problems. This methodology, which is independent of particular technologies or tools, with its ten stages shown in Fig. 1.3 should provide a framework for a data scientist in obtaining answers and results.

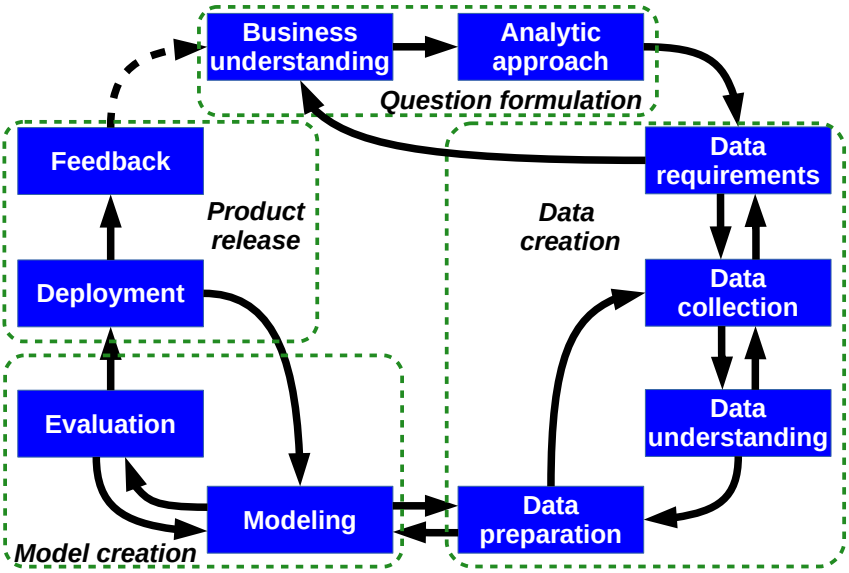


Fig. 1.3 Data science methodology according to John Rollins from IBM (with some slight additions)

Implementing machine learning models with scikit-learn is really easy with a few lines of Python code. Most of the tasks that need to be done in a machine learning pipeline (see data and model creation part in Fig. 1.3) are implemented already in scikit-learn including pre-processing of data, feature selection, feature extraction, train-test splitting, defining the algorithms, fitting models, tuning parameters, prediction, evaluation and exporting the model.

The pre-processing package of scikit-learn provides several common utility functions and transformer classes to change raw feature vectors into a suitable form of vector for modeling. There are also transformers for text data. Scikit-learn can split arrays or matrices into a random train and test subsets for you in one line of code so that you can train your model and then test the model's accuracy separately. Every machine learning algorithm in scikit-learn has two main methods. You can train your model bypassing your training set to the `fit` method of the selected algorithm, then you can use your test set to run predictions with the `predict` method, and use different metrics of scikit-learn to evaluate your model accuracy.

As you will see in the accompanying Jupyter Notebooks, all machine learning algorithms discussed in this book can be built using NumPy or SciPy packages. However, the entire process of a machine learning task can be done simply in a few lines of code using scikit-learn[15]. You should also check Appendix A for a very brief introduction to scikit-learn.

1.6.9 StatsModels

StatsModels library - started as part of SciPy but now is a standalone library - is a Python package that provides a complement to scipy for statistical computations including descriptive statistics and estimation and inference for statistical models[16]. The library is built on top of the numerical libraries NumPy and SciPy, integrates with Pandas for data handling and uses an R-like formula interface. Its graphical functions are based on the Matplotlib library.

Although scikit-learn library also contains regression, StatsModels library is an essential tool for those who work extensively with regression and time-series data. Although in the accompanying Jupyter Notebooks the NumPy and scikit-learn is the primary focus examples of using StatsModels are also provided.

1.7 Structure of the Book

This is the first of a series of the three-volume book titled "Advanced Machine Learning" dedicated for machine learning algorithms. The three volumes are as follows:

- Volume 1: Generalized Linear Models
- Volume 2: Supervised Learning

[15] Please, check https://scikit-learn.org

[16] For more information check https://www.statsmodels.org/stable/index.html

- Volume 3: Unsupervised Learning

The first volume - this one - is focusing mainly on linear models which are the most fundamental part of supervised learning. Although linear models have some limitations in their usage, they are simpler, thus easier to understand and shall be seen as a foundation in our understanding. This book also discusses data science methodology, basics of statistics, model assessment, and model validation which serves as a baseline for the other two volumes. The second volume discusses in more detail other supervised learning algorithms that are not part of generalized linear models. The third volume is solely dedicated to the unsupervised learning algorithms.

The second chapter of this book starts on clarifying the difference between the population and sample, and what statistical measures are used to characterize a population and how these statistical measures can be estimated from the sample. Then the rest of the chapter focuses on the simple linear regression model: what are the mathematical foundations of the linear model fit, what metrics we can use to check the goodness of the model, and what uncertainties are related to the predictions of the model due to sampling error. The assumptions of the linear model fit are also discussed, as well as some methods to use for checking the correctness of these assumptions. At the end of this chapter outliers and influential points are also discussed. The chapter closes with emphasizing the importance of graphing and the usefulness of data pre-processing.

The third chapter discusses the mathematical framework of the multiple linear regression, deriving all the mathematical formulas for solving multiple linear regressions problems, respective how outliers and influential points can be suspected in such cases. Then the chapter continues discussing some special cases of linear regression, like polynomial regression, weighted least square regression, ridge and Lasso regression, robust regression and Bayesian regression. This chapter also discusses in more detail the data pre-processing techniques, data visualization, respective how discrete input variables shall be handled. A great emphasis on learning theory and generalization capabilities of multiple linear regressions are also discussed.

The fourth chapter is dedicated to the time series analysis as a special case of linear regressions. It starts with introducing the time series, what is the fundamental difference compared to other data. Then different time series modeling is presented. The chapter ends with the discussion of multiple time series.

The fifth chapter is a brief introduction to some optimization methods, focusing on the stochastic gradient descent, Newton method, and iteratively reweighted linear least squares. Stochastic gradient descent is a very important optimization method which is also used in neural networks and deep learning. Thus, understanding its usage can be used

The sixth chapter discusses in detail the logistic regression, a form of a linear classifier. It starts by formulating the problem of classification and the relation to linear regression. Then binary and multi-class classification is discussed in detail. This shall be seen as a foundation in classification problems that can be used later on.

Chapter seventh introduces into the realm of non-linear regression with kernel methods which is still bases on the linear regression. The last chapter discusses generalized linear models and generalized additive models.

References

1. Aghabozorgi, S., Machine Learning with Python, IBM, Coursera, 2018. https://www.coursera.org/learn/machine-learning-with-python/
2. Aklson, A., What is Data Science?, IBM, Coursera, 2018. https://www.coursera.org/learn/what-is-datascience
3. Aklson, A., Data Science Methodology, IBM, Coursera, 2018. https://www.coursera.org/learn/data-science-methodology
4. Brooks, C., Introduction to Data Science in Python, University of Michigan, Coursera, 2017. https://www.coursera.org/learn/python-data-analysis/
5. Conway, D., The Data Science Venn Diagram, 2015. http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram
6. Dohono, D., 50 years of Data Science, Princeton NJ, 2015. http://courses.csail.mit.edu/18.337/2015/docs/50YearsDataScience.pdf
7. Ericson, G., et al, What is the Team Data Science Process?, Microsoft Azure, 2017. https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/overview
8. Faggella, D., What is Machine Learning?, Techemergence, 2018. https://www.techemergence.com/what-is-machine-learning/
9. Heider, M., Getting Started with Data Science: Making Sense of Data with Analytics, IBM Press, 2015.
10. Hunter, J. D., Matplotlib: A 2D graphics environment, Computing In Science & Engineering, vol. 9, nr. 3, pp. 90–950, 2007, COMPUTER SOC doi: 10.1109/MCSE.2007.55. https://ieeexplore.ieee.org/document/4160265
11. Jones E, et al., SciPy: Open Source Scientific Tools for Python, 2001. http://www.scipy.org/
12. Moor, J., The Dartmouth College Artificial Intelligence Conference: The Next Fifty years, AI Magazine, Vol 27, No. 4, Pp. 87-89, 2006. https://pdfs.semanticscholar.org/d486/9863b5da0fa4ff5707fa972c6e1dc92474f6.pdf
13. Ng, A., Machine Learning and AI via Brain Simulations, Stanford University, Course slides, 2013. http://ai.stanford.edu/~ang/slides/DeepLearning-Mar2013.pptx
14. Ng, A., Machine Learning, Stanford University, Coursera, 2012. https://www.coursera.org/learn/machine-learning
15. Palmer, S, Data Science Advisory. https://www.shellypalmer.com/data-science/
16. Pedregosa F. et al., Scikit-learn: Machine Learning in Python, JMLR 12, pp. 2825-2830, 2011. http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html
17. Pérez, F., Granger, B. E., IPython: A System for Interactive Scientific Computing, Computing in Science and Engineering, vol. 9, no. 3, pp. 21-29, May-June 2007 doi: 10.1109/MCSE.2007.53 https://ipython.org
18. Polong, L., Open Source tools for Data Science, IBM, Coursera, 2018. https://www.coursera.org/learn/open-source-tools-for-data-science
19. Rollins, J. B., Foundational Methodology for Data Science, White Paper, IBM Analytics, 2015. https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=IMW14824USEN

20. Seabold, S., et al., Statsmodels: Econometric and statistical modeling with python, Proceedings of the 9th Python in Science Conference, 2010.
http://conference.scipy.org/proceedings/scipy2010/pdfs/seabold.pdf
21. Shmuelli, G., To explain or to Predict?, Statistical Science, Vol 25, No. 3, Pp. 289-310, 2010.
https://www.stat.berkeley.edu/~aldous/157/Papers/shmueli.pdf
22. Venners, B., The Making of Python: A Conversation with Guido van Rossum, Artima, 2003.
https://www.artima.com/intv/pythonP.html
23. Wu, C. F. J., Statistics = Data Science?, Georgia Tech, 1997.
http://www2.isye.gatech.edu/~jeffwu/presentations/datascience.pdf

Chapter 2
Simple Linear Regression

2.1 Introduction

Linear regression represents a linear approach for modeling the relationship between input(s) and output. The inputs go by different names, such as ***predictors***, ***regressors***, ***covariates***, ***features***, ***independent variables***, or ***explanatory variable***, and are denoted using the symbol x for a single input (or $\mathbf{x}$ for multiple inputs). The output variable is often called the ***response***, ***regressand***, ***target***, ***label*** or ***dependent variable***, and is denoted using the symbol y (or $\mathbf{y}$ for multiple outputs). Throughout this book, we will use all of these terms interchangeably[1].

Linear regression was the first type of regression analysis to be studied rigorously and to be used extensively in practical applications. This is because models that depend linearly on their unknown parameters are easier to fit than models that are non-linearly related to their parameters. In the next coming section, we will clarify the meaning of a model parameter. The case of a single explanatory variable (one single input) is called ***simple linear regression***.

2.2 Case Study: Is Height Hereditary?

The term "regression" was coined by Francis Galton[2] in the nineteenth century to describe a biological phenomenon. He observed that extreme characteristics (e.g., height) in parents are not passed on completely to their children. Rather, the characteristics in the children regress towards a mediocre point (i.e. the mean). More specifically, the heights of descendants of tall ancestors tend to ***regress*** down to-

[1] In statistics and econometrics the term ***endogenous variable*** is also used for the dependent variable which values are determined by other (independent) variables in the system, the latter variables called the ***exogenous variables***. Statsmodels library also uses these terms for input and output variables.

[2] Sir Francis Galton was an English polyhistor; he was Victorian-era statistician, polymath, sociologist, psychologist, anthropologist, eugenicist, tropical explorer, geographer, inventor, meteorologist, proto-geneticist, and psychometrician.

wards a normal average (a phenomenon is also known as regression toward the mean).

By measuring the heights of hundreds of people Galton was able to quantify regression to the mean, and estimate the size of this effect which he called the "law of regression". He wrote:

> "... we can define the law of regression very briefly. It is that the height-deviate of the offspring is, on the average, two-thirds of the height-deviate of its mid-parentage." [10]

In other words, the size of the deviation of a child's height from the mean on average is only 2/3 of the deviation of parents' mid-height from the average height of adults. Today we say that the regression coefficient (as "the law of regression") tells us that for one additional unit increase (or decrease) in parents' mid-height would result in (approximately) 2/3 additional increase (or decrease) in children's height in average. You can see the scatter plot on the left of Fig. 2.1 which was created from the historical dataset `GaltonFamilies`[3] (you may also check [14] for an in-depth analysis of this dataset). On this plot, each point represents a pair of parents' mid-height and their child's height. The regression coefficient ("the law of regression") is the slope of the blue line in Fig. 2.1, the latter called the ***regression line***.

Now let's see, what the regression line represents and how it can be defined? In this section only the first part of the question will be answered, the mathematical definition of the linear regression will be provided in the upcoming sections. To construct an intuition for answering the first part of the question, let's define 11 intervals on parents' mid-height, each having a length of approximately 1 inch. You can see this in Fig. 2.2 where for each interval a different color is used when plotting the pairs of data.

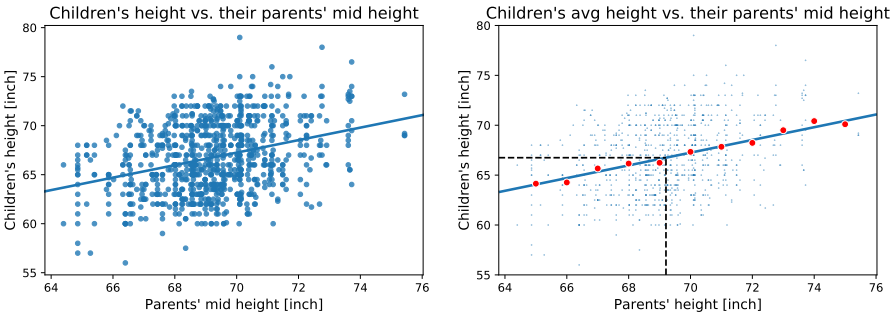


Fig. 2.1 Scatter plot of the `GaltonFamilies` dataset. Each point on the left represents one observation from the dataset with the parents' mid-height in the horizontal axis and their children's height on the vertical axis. The blue line shows the regression line. The red points on the right represent the average height of children defined over an approximately 1-inch interval of parents' mid-height.

[3] `GaltonFamilies` dataset is available in the `Hist` package of the R language.

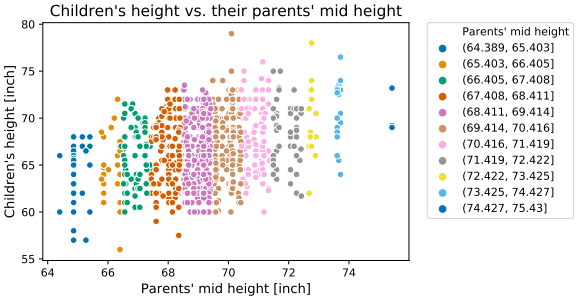


Fig. 2.2 Scatter plot of the `GaltonFamilies` dataset using different colors for groups of observations. At this time the parents' mid height is divided into eleven intervals - each interval being approximately 1 inch wide - and for each interval a different color is used for the data points.

Next, we calculate the average height of the children in each interval and make a scatter plot of these values together with the regression line defined over the entire dataset. As we can observe on the right of Fig. 2.1 these average values of children's height are falling almost on the regression line. Thus, we may conclude that the regression line represents the average value of the children's height for any particular value of their parents' mid-height. With dashed lines, we also mark the average value (mean) of the parents' mid-height, respective children's mid-height. Then any additional increase (or decrease) - as a deviation from the mean - of the parents' mid-height would result in approximately 2/3 increase (or decrease) - as a deviation from the mean - of the children's height.

As we will see later in this chapter (Sec. 2.6.2) a score value can be defined for the regression model which could be between 0 and 1, and the higher the score value the better the model fit on the dataset. In this case, the score value of the regression model is only 0.103 which is very low suggesting that important information is missing from our model. As we can observe in the scatter plot shown on the left of Fig 2.1 there is much variability in the children's height. The correlation coefficient (Sec. 2.5.2.4) tells us how much two random variables are varying together. In this case, the correlation coefficient between the parents' mid-height and children's height is only approximately 1/3, thus 2/3 of the children's height variability cannot be explained by the parents' mid-height, this is related to some other factors.

We expect that male children on average are taller than female children. Thus, we can improve our regression model by taking into account other factors, like the gender of the children. In such a case we enter the realm of multiple linear regression discussed entirely in Chapter 3. The gender of the child has a much higher correlation coefficient with the child height than the ancestor's mid-height, namely 0.72. If we take both factors into account then there will be two regression lines (parallel to each other) as shown on the left of Fig. 2.3, one regression line for each gender. The score of this regression model is much higher, namely 0.633. We may try further improvement of our model by taking not the mid-height of the parents, but rather the height of the father and mother. However, the result, in this case, will not be two regression lines, but rather two parallel ***regression plane***s (one for each gender) as shown on the right of Fig. 2.3. When calculating the score of this model we find out that only a very-very tiny improvement could be achieved, namely, the score value becomes 0.635.

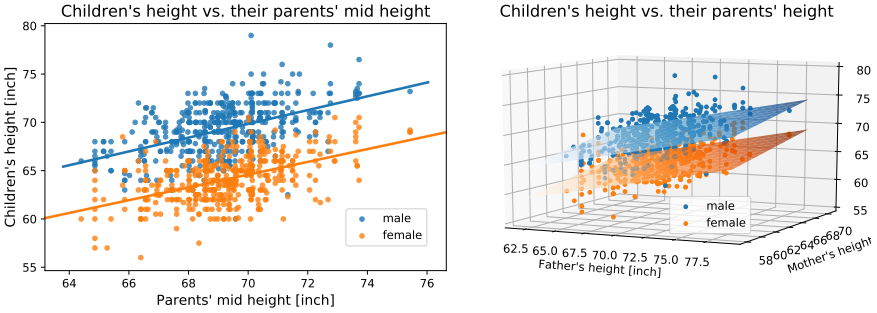


Fig. 2.3 Scatter plot of the `GaltonFamilies` dataset where the gender of the child is also counted. Depending on whether we use only the parents' mid height or we are using both the father's and mother's height we will get two parallel regression lines (on the left) or two parallel regression planes (on the right).

Now, that we know what the regression line or plane represents we could also have an idea about the usefulness of it. If we know the regression line or plane we can make a prediction about the average height of children based on their parents' height. In the upcoming sections, we will build up the mathematical framework to understand deeply the linear regression model and what uncertainties are related to our prediction based on the regression line (or plane).

2.3 Simple Dataset with One Explanatory Variable

To have a clear understanding of machine learning algorithms some notions used in this book series shall be clarified first. For this reason, let's take a very simple dataset of systolic blood pressure of peoples of different ages. Our goal is to analyze if there is a linear relationship between the age of the people and their systolic blood pressure. In this case, the age of the people represents the input variable and all the values that the input can take (i.e. all possible ages) the input space denoted with $\mathcal{X}$. The systolic blood pressure represents the output variable and all the values that can take the output space denoted with $\mathcal{Y}$.

In Table 2.1 a part of the `BloodPressure1` dataset is shown where the input variable (`Age`) is denoted by x and the output variable (`Systolic blood pressure`) by y. This is an example of a ***structured data***, where data is organized in a table: a column of the table represents the values of an input or output, while a row - also called an entry - represents a single observation. This dataset consists of $N = 30$ pairs of (x, y) values which are called ***data points***, ***observations***, or ***samples***.

To distinguish between the data points a superscript index in bracket is also applied to each input and output value. Brackets are added to the index to distinguish from the power to N. Superscript is applied instead of a subscript to not confuse the training example indexing with the feature indexing. That is when there is more than one input variable (more than one feature) subscript indexing is also applied to the features, as we will see later on (see Table 3.1 for feature indexing). Moreover,

class indexing, as another subscript, will be also applied in classification algorithms. These are the main reasons that superscript instead of subscript indexing is used for identifying observations[4].

Table 2.1 Systolic blood pressure as function of age

x	Age	y	Systolic blood pressure
$x^{(1)}$	39	$y^{(1)}$	144
$x^{(2)}$	47	$y^{(2)}$	220
$x^{(3)}$	45	$y^{(3)}$	138
$x^{(4)}$	47	$y^{(4)}$	145
$x^{(5)}$	65	$y^{(5)}$	162
$x^{(6)}$	46	$y^{(6)}$	142
$x^{(7)}$	67	$y^{(7)}$	170
$x^{(8)}$	42	$y^{(8)}$	124
$x^{(9)}$	67	$y^{(9)}$	158
.....	.....	.....	.....
$x^{(N)}$	69	$y^{(N)}$	175

Source: http://people.sc.fsu.edu/~jburkardt/datasets/regression/regression.html

This dataset - denoted by $\mathcal{D}_N = \{\{x^{(1)}, y^{(1)}\}, \{x^{(2)}, y^{(2)}\}, ..., \{x^{(N)}, y^{(N)}\}\}$ - is rather a toy example compared to the huge datasets available nowadays, but let's make our first baby steps using this simple one. As we get more confident in the mathematical notations and statistical concepts more complex problems will be analyzed.

Note *As we will see in later chapters, as long as the number of input variables is much lower than the number of observations available, the same algorithms can be used whether we have only a few hundred or a few millions of observations. The only difference might be the underlying HW and SW technology used to run the machine learning algorithm.*

One of the first actions in analyzing a dataset is to create a graphical representation of the observations, as shown in Fig. 2.4. Some examples in Sec. 2.12 are provided to understand the importance of graphing.

The scope of our investigation is to find out if there is a linear relationship between blood pressure and the age of a person. Before proceeding, we must clarify what types of relationships we won't study in this book, namely, deterministic relationships; like in case of converting from degrees Celsius to degrees Fahrenheit or calculating the circumference of the circle based on its diameter. For each of these deterministic relationships, also called functional relationship and denoted by $y = h(x)$, there is an equation exactly describing the relation between the two variables.

Instead, we are interested in statistical relationships, in which the relation between the variables is not perfect. Looking at the scatter plot in Fig. 2.4 it is obvious

[4] This follows Andrew Ng's notation of Machine Learning class from Stanford University.

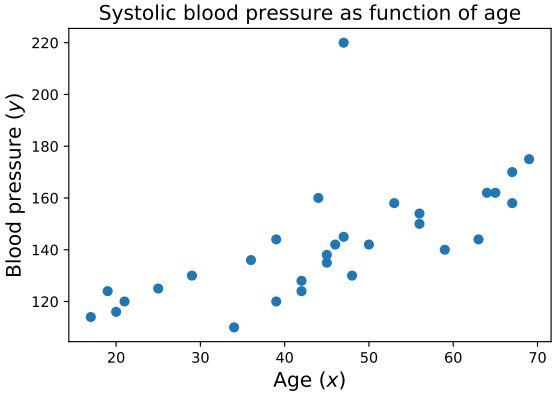


Fig. 2.4 Systolic blood pressure as the function of the age of a person

that there is no deterministic linear relation between the age and blood pressure because the observations cannot be placed perfectly on a single line. But before continuing our analyzes let's recap some of the notions related to probability theory.

2.4 Random Variables

Probability theory is about random variables (denoted with italic capital letters, like X), where a random variable can be thought of as an uncertain, numerical quantity (i.e. a continuous random variable takes values in $\mathbb{R}$) resulting from a random process, also called ***stochastic process***. While we do not know with certainty what value a random variable X will take, we usually know (or at least we try to estimate) how to compute the probability - i.e. a measure of taking values between 0 and 1 and denoted with $\mathbb{P}(X)$ -, that its value will be in some subset of $\mathbb{R}$. The set of values (or outcomes) of a random variable is called an event.

The collection of all probabilities of a random variable is called the distribution of X. One has to be very careful not to confuse the random variable itself and its distribution. The sum of all those probabilities must be equal to 1 because the probability of a certain event is 1. On the other hand, an impossible event has probability 0.

2.4.1 Discrete Random Variables

A random variable is said to be discrete if it takes only values from a countable set[5]. When rolling a fair die, we have a stochastic process with possible outcomes of $\{1,2,3,4,5,6\}$, so this is an example of a discrete random variable. The probability of each value from this set of $\{1,2,3,4,5,6\}$ is $\frac{1}{6}$, and the distribution, called

[5] A countable set is either a finite set or a countably infinite set. Whether finite or infinite, the elements of a countable set can always be counted one at a time and, although the counting may never finish, every element of the set is associated with a unique natural number. (https://en.wikipedia.org/wiki/Countable_set).

probability mass function (PMF) is

$$P(X) = \Big\{\mathbb{P}(X=1) = \frac{1}{6}, \mathbb{P}(X=2) = \frac{1}{6}, \mathbb{P}(X=3) = \frac{1}{6}, \\ \mathbb{P}(X=4) = \frac{1}{6}, \mathbb{P}(X=5) = \frac{1}{6}, \mathbb{P}(X=6) = \frac{1}{6}\Big\} \tag{2.1}$$

and we call it a uniform distribution. The probability to obtain any value from the set of $\{1,2,3,4,5,6\}$ is 1 (it is a certain event), and the probability of any value outside of this set is 0 (impossible event).

Note

A more tangible way to grasp the notion of the probability of an event (like rolling a die to get a 6) is to think as a proportion of times the event occurs when we repeat the experiment over and over independently and under the same conditions. We use the notation $\mathbb{P}(A)$ to denote the probability of an event A happening. We use the very general term event to refer to things that can happen by chance.

Another example to a discrete random variable is when tossing a fair coin, where the outcome is either "head" or "tail" with probability of 0.5 for each event, and the probability mass function is $P(X) = \{\mathbb{P}(\text{Head}) = 0.5, \mathbb{P}(\text{Tail}) = 0.5\}$. This is an example of uniform ***Bernoulli distribution*** which takes the value 1 (set for "Head", for example) with probability p and the value 0 (set for "Tail") with probability $q = 1 - p$ with equal chance (i.e. $p = q = 0.5$).

Now, if we toss two fair coins at once then we can create a ***joint distribution*** of these two discrete random variables X and Y, which defines probabilities for each pair of outcomes. Since each outcome is equally likely the joint probability mass function becomes the one shown in Table 2.2.

Table 2.2 The joint distribution of two discrete random variables (tossing two fair coins at once)

Joint distribution table		X **(Coin #1)**	
		Head (1)	**Tail (0)**
Y **(Coin #2)**	**Head (1)**	0.25	0.25
	Tail (0)	0.25	0.25

The ***conditional probability*** of X given Y - denoted with $\mathbb{P}(X|Y)$ - is the probability of X when the value of Y is known (i.e. is already observed). For example, $\mathbb{P}(X = 1|Y = 1) = 0.25$ and the ***conditional distribution*** of X given $Y = 1$ is represented by the first row in Table 2.2.

The ***marginal distribution*** of a random variable is the probability distribution of the variable (the probabilities of various values of the variable) without reference to the values of the other variable. From Table 2.2 we can obtain the marginal distribution of X if we sum up the values in the columns (obtaining $\{\mathbb{P}(\text{Head}) = 0.5, \mathbb{P}(\text{Tail}) = 0.5\}$), and we can obtain the marginal distribution of Y if we sum up the rows (obtaining $\{\mathbb{P}(\text{Head}) = 0.5, \mathbb{P}(\text{Tail}) = 0.5\}$). Of course, the sum of all probabilities in the table should be equal to 1.

Two events are ***statistically independent*** if the occurrence of one does not affect the probability of occurrence of the other. In other words, two events are independent if the outcome of one does not affect the other. Then two random variables are said to be independent if the realization of one does not affect the probability distribution of the other. Tossing two dies is an example of independent random variables. In such cases, the joint probability mass function is the product of the marginals (like we have in Table 2.2).

$$\mathbb{P}(X,Y) = \mathbb{P}(X)\mathbb{P}(Y) \quad \text{for} \quad X,Y \in \{\text{Head}, \text{Tail}\} \tag{2.2}$$

that is

$$P(X) = \begin{cases} \mathbb{P}(\text{Tail}, \text{Tail}) = 0.5 \cdot 0.5 = 0.25 \\ \mathbb{P}(\text{Tail}, \text{Head}) = 0.5 \cdot 0.5 = 0.25 \\ \mathbb{P}(\text{Head}, \text{Tail}) = 0.5 \cdot 0.5 = 0.25 \\ \mathbb{P}(\text{Head}, \text{Head}) = 0.5 \cdot 0.5 = 0.25 \end{cases} \tag{2.3}$$

We can also define the ***cumulative distribution function*** (CDF) - denoted with $F(X)$ - which answers the question that what proportion of the values of a random variable is less than or equal to a given value x, i.e. $F(x) = \mathbb{P}(X \leq x)$[6]. For example, throwing a fair die has a cumulative distribution function:

$$F(X) = \begin{cases} \mathbb{P}(X \leq 1) = \dfrac{1}{6} \\ \mathbb{P}(X \leq 2) = \dfrac{1}{3} \\ \mathbb{P}(X \leq 3) = \dfrac{1}{2} \\ \mathbb{P}(X \leq 4) = \dfrac{2}{3} \\ \mathbb{P}(X \leq 5) = \dfrac{5}{6} \\ \mathbb{P}(X \leq 6) = 1 \end{cases} \tag{2.4}$$

If we have N number of independent coin flips then we have the case of a ***binomial distribution***. In general, if the random variable X follows the binomial distribution with parameter $p \in [0,1]$ and k then the probability of getting exactly k "heads" in N independent Bernoulli trials is given by the PMF:

$$f(p,N,k) = \mathbb{P}[X = k] = \binom{N}{k} p^k (1-p)^{N-k} \tag{2.5}$$

The formula can be understood as follows: k "heads" occur with probability p^k and $N-k$ "tails" occur with probability $(1-p)^{N-k}$. However, the k "heads" can occur

[6] Remember, X denotes the random variable, while x denotes the observed value of the random variable X.

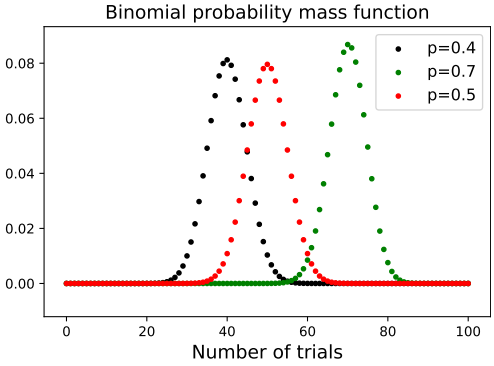


Fig. 2.5 PMF plot of a binomial distribution representing $N = 100$ coin flips using using three different coins: one is fair ($p = 0.5$), the other two are biased ($p = 0.4$ biased slightly toward "tail', while $p = 0.7$ biased more to the "head").

anywhere among the N trials, and there are $\binom{N}{k}$ different ways of distributing k "heads" in a sequence of N trials. In Fig. 2.5 the PMF of $N = 100$ coin flips are provided using three different coins: one is fair ($p = 0.5$), the other two are biased ($p = 0.4$, respective $p = 0.7$).

2.4.2 Continuous Random Variables

A continuous random variable is one that can take on infinitely many, uncountable values. For example, a variable over a non-empty range of the real numbers is continuous, if it can take on any value in that range. The distribution of a continuous random variable is called the ***probability density function*** (PDF) denoted with $p(X)$ which is a continuous function over the range in which X can take values. There are several well-known distributions, but the most important is the normal distribution, so our focus is placed on this in the next section.

2.4.2.1 Normal Distributions

Normal distributions - also called ***Gaussian distribution*** or simply a ***bell curve*** due to its shape - are important in statistics and are often used in the natural and social sciences to represent real-valued random variables whose distributions are not known. Univariate normal distribution with means μ and variance σ^2 is denoted by $\mathcal{N}(\mu, \sigma^2)$ and it is defined completely by these two numbers. The PDF of the univariate normal distribution with zero mean ($\mu = 0$) and unit standard deviation ($\sigma = 1$) is shown on the left of Fig. 2.6 where each band has a width of 1 standard deviation. The 68-95-99.7 rule, also known as the empirical rule, is a shorthand used to remember the percentage of values that lie within a band around the mean in a normal distribution with a width of two ($\pm\sigma$), four ($\pm 2\sigma$) and six ($\pm 3\sigma$) standard deviations, respectively (see the bends on the left of Fig. 2.6). More accurately, 68.27%, 95.45% and 99.73% of the values lie within one, two and three standard deviations of the mean, respectively. The main message of the bell curve is that most of the values of the population lie around its mean.

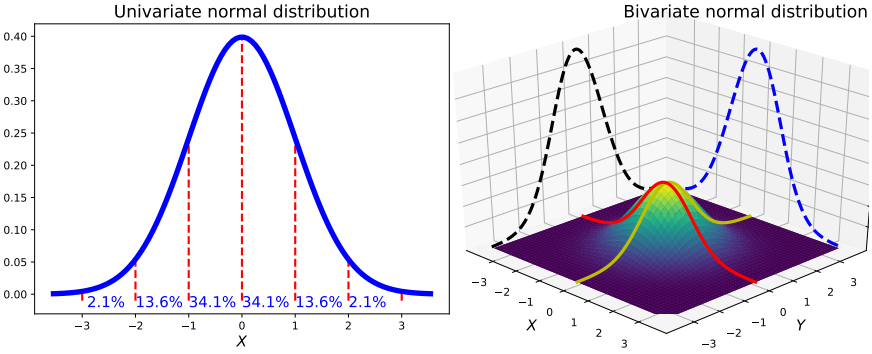


Fig. 2.6 PDF plot of a normal distribution (or bell-shaped curve) on the left where each band has a width of 1 standard deviation, and a plot of a bivariate normal joint density surface on the right (where the bell shape curves along the surface represent the conditional probability distributions of Y when $X = 0$ (yellow), respective of X when $Y = 0$ (red); while the dashed bell curves on the "walls" represent the marginal probabilities of X (blue), respective Y (black)).

Note

Physical quantities that are expected to be the sum of many independent processes (such as measurement errors) often have distributions that are nearly normal (see Sec 2.4.2.2). We seem to be surrounded by bell curves. All manner of things appear to be distributed normally: people's heights, sizes of snowflakes, errors in measurements, lifetimes of lightbulbs, IQ scores, weights of loaves of bread, and so on [17].

The cumulative distribution function of a real-valued random variable X, evaluated at x, is the probability that X will take a value less than or equal to x. In the case of a continuous distribution, it gives the area under the PDF from minus infinity to x. In the case of the normal distribution, the CDF starts from zero and increases steadily until one as shown in Fig. 2.7.

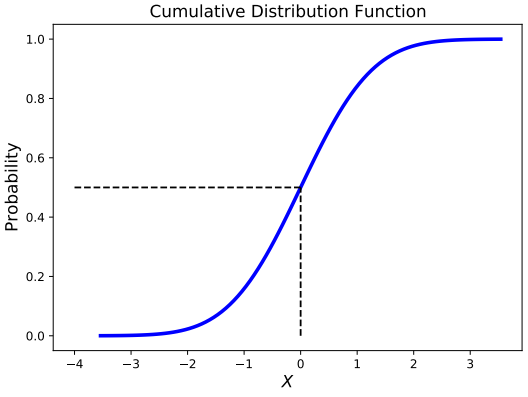


Fig. 2.7 Cumulative distribution function (CDF) of a standard normal distribution with zero mean (the probability of X being less than or equal to zero is 0.5)

You can only use r to make a statement about the strength of the linear relationship between x and y, because correlation does not mean causality! In plain words, when x is strongly correlated with y does not imply that the change in x causes a change in y. It simply means, that when there is a change in x there is some change in y, and vice versa.

For example, alcohol and lung cancer are correlated, but this is only because drinkers also tend to be smokers (alcohol does not biologically cause lung cancer) [22]. You can see many more such absurd examples on the Spurious Correlations website[14].

Sample correlation coefficient r is an estimate of ρ and used to measure the strength of the linear relationship in the sample. Because is calculated using N random observations, the sample correlation coefficient represents a random variable. In general:

- If $r_{xy} = -1$, then there is a perfect negative linear relation between x and y.
- If $r_{xy} = +1$, then there is a perfect positive linear relation between x and y.
- If $r_{xy} = 0$, then there is no linear relation between x and y.

All other values of r_{xy} tell us that the linear relation between x and y is not perfect. The closer r_{xy} is to 0, the weaker the linear relationship. The closer r_{xy} is to -1, the stronger the negative linear relationship, and, the closer r_{xy} is to $+1$, the stronger the positive linear relationship.

In Fig. 2.11 some correlation examples are shown. The plot on the right bottom is an example when the error spread is not constant and depends on the value of x. This is called ***heteroscedasticity***. The other three plots are examples of errors with constant variance, called ***homoscedasticity***.

2.6 The Simple Linear Regression Model

If we examine our small dataset presented in Table 2.1 we can observe that entry with index 2 and 4 both have the same input (age 47) while the outputs are different (220 vs 145). Similarly, we can see that entries with index 7 and index 9 both have the same age of 67, while the blood pressure is 170 units in one case, and 158 units in the other. That should not be a surprise! There are a lot of people on the Earth with age 47 or 67, but we should not expect that everybody would have the same blood pressure corresponding to age 47 or 67. There are many factors besides the age which could affect a person's blood pressure: weight, stressful life, the side effect of drugs, illness, etc. In our study of simple linear regression, we are only focusing on age as the only explanatory variable.

If we were to measure the blood pressure of every people having age 47 then we would have a lot of different blood pressure values that are spread around a mean

[14] See http://tylervigen.com/spurious-correlations

value. That is, the measured blood pressure of different persons with the same age would have a (normal) distribution around the mean corresponding to that age and for each age, we expect to have a different mean (after all, we are interested in finding the linear relation between the age and blood pressure).

This mean value is called ***conditional mean*** of y given x and denoted by $\mu_{y|x}$, because we are interested for the mean of y only for a given x. In order to have a better understanding, let's visualize this by having for an input value $x^{(i)}$ the corresponding output values $y^{(i1)}, y^{(i2)}, \ldots$ with a normal distribution around its conditional mean $\mu_{y|x^{(i)}}$. Similarly, we can have another input value $x^{(j)}$ with corresponding output values $y^{(j1)}, y^{(j2)}, \ldots$ with a normal distribution around its conditional mean $\mu_{y|x^{(j)}}$. See Fig. 2.17 for the two bell shape distribution of y around the conditional mean values corresponding to the two inputs mentioned before.

Of course, the same can be done for each age in interest - or for each x values, in general -, and calculate the conditional mean for each age (conditional mean of y). If there is a linear relationship between the age of a person and the corresponding average blood pressure then the conditional mean values would lie on a single line. This is called the ***population regression line*** which summarizes the trend in the population between the predictor x and the mean of the responses $\mu_{y|x}$. We can also define an ***error*** term $\varepsilon^{(i)}$ between the actual observed output $y^{(i)}$ and its conditional mean $\mu_{y|x^{(i)}}$.

By the nature of the problem we can state that the input is not a random variable (we assume it can be measured without error) while the output is a random variable forming a distribution around the corresponding conditional mean. Based on how we defined the error term we can also conclude that the error term is also a random variable. Then we can write:

$$\begin{aligned} \mu_{y|x} &= \beta_1 \cdot x + \beta_0 \\ y^{(i)} &= \underbrace{\beta_1 \cdot x^{(i)} + \beta_0}_{\text{deterministic}} + \underbrace{\varepsilon^{(i)}}_{\text{random}} \quad \text{for} \quad i = 1, 2, \ldots \end{aligned} \tag{2.39}$$

where β_0 represents the intercept of the population regression line (when $x = 0$), while β_1 measures the average change in the dependent variable as the independent variable changes one unit. The population attributes β_1, β_0 are called the ***regression coefficients***.

In Fig. 2.12 some artificial observations are shown for a population with empty and filled bubbles where the red dashed line represents the population regression line. The term $\varepsilon^{(i)} = y^{(i)} - \mu_{y|x^{(i)}}$ denotes the error between the population regression line (conditional mean) and the actual observation and represents the unpredictable random error. As we stated earlier $\varepsilon^{(i)}$ error is due to factors outside of the age (like weight, stressful life, the side effect of drugs, or even measurement or typing error, etc.).

Taking into account our assumption about the input variable being deterministic and the output variable[15] together with the unpredictable error being random we can also derive from Eq. 2.39:

$$\begin{aligned}\mathbb{V}\left[y^{(i)}\right] &= \mathbb{V}\left[\beta_1 \cdot x^{(i)} + \beta_0\right] + \mathbb{V}\left[\varepsilon^{(i)}\right] = \sigma_\varepsilon^2 \\ \mathbb{E}\left[y^{(i)}\right] &= \mathbb{E}\left[\beta_1 \cdot x^{(i)} + \beta_0\right] + \mathbb{E}\left[\varepsilon^{(i)}\right] = \beta_1 \cdot x^{(i)} + \beta_0\end{aligned} \tag{2.40}$$

where the deterministic part plays the role of a constant, so its variance is zero. We also assume that the random error has zero mean (see Sec. 2.7.2) and constant variance σ_ε^2 (see Sec. 2.7.5).

So far so good. However, conducting an experiment during which the blood pressure of all the people on the entire planet is measured would be very-very expensive. Thus, we have to rely on a random sample of this population (a subgroup of the population like the filled bubbles shown in Fig. 2.12 or the small dataset presented in Table 2.1). Then the aim is to find a linear relationship between the age and blood pressure of people based on the observations in the sample dataset. That would result in drawing a line through the observations and based on that prediction could be made. A line would have an ***intercept*** and a ***slope***, which are called the unknown ***parameters*** of the linear regression model. To stick to the naming convention and notations used in neural networks, the intercept will be denoted with b and called ***bias***[16], while the slope will be denoted with w and called ***weight***. With these notations we can write:

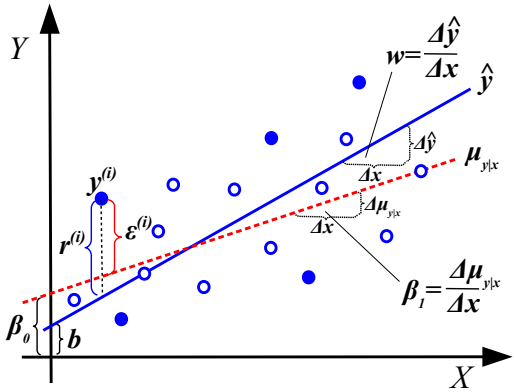


Fig. 2.12 Population regression line (dashed red) vs. estimated regression line (continuous blue)

[15] If the output is a random variable with a distribution around its conditional mean then a random sample $y^{(i)}$ from this distribution also represents a random variable.

[16] This bias - representing the intercept - should not be mixed up with the bias of the estimator, i.e. the parameters of the linear model. While the model parameter b as an estimator of β_0 could have zero bias (i.e. $\mathbb{E}[b]=\beta_0$) the bias itself could be nonzero (i.e. $b \neq 0$). To avoid confusion the term intercept or constant will be used in linear models while still denoting with b.

$$\hat{y} = w \cdot x + b$$
$$y^{(i)} = w \cdot x^{(i)} + b + r^{(i)} \quad \text{for} \quad i = \overline{1,N} \tag{2.41}$$

where $\hat{y}$ symbolizes that this is only the estimate and not the real conditional mean $\mu_{y|x}$ of the population and N represents the sample size. The line defined by Eq. 2.41 is called the ***estimated regression line*** and shown with straight blue line in Fig. 2.12. This line is estimated based on the sample (randomly selected observations from the population shown with the filled bubble), and not on the whole population.

The bias b (also called ***constant***) is an estimator of the population attribute β_0, while the weight w (also called ***coefficient***) is an estimator of the population attribute β_1. The term $r^{(i)} = y^{(i)} - \hat{y}$ represents the ***residual*** (or error[17]) between the estimated regression line and the actual observation and is an estimate of the true error $\varepsilon^{(i)}$.

Note

The residuals $r^{(i)}$ are not the true errors $\varepsilon^{(i)}$, but estimates, based on the observable data in the sample. The closer the estimated regression line to the population regression line the better the estimation of error terms.

The residual occurs because - as we stated earlier - there is no deterministic linear relationship between the age of a person and his/her blood pressure. There will be always a residual between the estimation (also called ***prediction***) and the actual observation. Now, if we sum up the second equality of Eq. 2.41 for all N and divide by N each side we obtain:

$$\frac{1}{N}\sum_{i=1}^{N} y^{(i)} = w\frac{1}{N}\sum_{i=1}^{N} x^{(i)} + b\frac{1}{N}\sum_{i=1}^{N} 1 + \frac{1}{N}\sum_{i=1}^{N} r^{(i)} \quad \Rightarrow \quad \bar{y} = w\bar{x} + b \tag{2.42}$$

where $\bar{x}$ and $\bar{y}$ represents the average of the input, respective output values from the sample. Similarly we can obtain $\bar{y} = \beta_1\bar{x} + \beta_0$ using Eq. 2.39 and the assumption that the error ε has zero mean. Thus, the population regression line will pass through the point (μ_x, μ_y) while the estimated regression line will pass through the point $(\bar{x}, \bar{y})$. It should be observed, that the two points are not equivalent in general (in most cases $\bar{x} \neq \mu_x$ and $\bar{y} \neq \mu_y$).

2.6.1 Fitting the Best Line

Now, that we clarified that we are aiming to draw a line through the observations of the sample let's, find a function through the ***learning*** process which could fit the best line through the observations and make good prediction for the blood pressure of a person not contained in the dataset (so-called new observation). Formally, the learning problem is to find out a function $f : \mathcal{X} \to \mathcal{Y}$ such that $f(x) = \hat{y}$ is a good predictor for the values of y. Here $\mathcal{X}$ represents the input space (in our example this is the age of the person, and could be between 0-100), while $\mathcal{Y}$ represents the output space (e.g. the possible values for systolic blood pressure). The function $f(x)$

[17] In the book the term residual will be used for the error between observation and estimated regression line in order to distinguish it from the true error ε.

is called the ***hypothesis*** because one never can be sure that the obtained function through learning will capture exactly the underlying association between the input and the output.

Of course, one could draw dozens of lines going through the scatter plot of the data points shown in Fig. 2.4. Thus, the goal of the learning process is to find the unknown parameters (weight w and bias b) such, that the obtained linear function would make the "best" prediction among all other competing options (all other straight lines). But, how do we find the "best" regression line?

A good approach would be to minimize the residuals:

$$r^{(i)} = y^{(i)} - \hat{y}^{(i)} \quad \text{for} \quad i = \overline{1,N} \tag{2.43}$$

Summing up the residuals would not be an option because the residual would be positive for some observations and negative for others (see Fig 2.13) so they would mostly cancel each other out . Summing the absolute values of the residuals would be a better approach, but the absolute function is not derivable at zero. This makes hard to find the minimum. Instead, the square of individual residuals shall be summed which is called the ***residual sum of squares*** (RSS):

$$\text{RSS} = \sum_{i=1}^{N} (r^{(i)})^2 = \sum_{i=1}^{N} (y^{(i)} - \hat{y}^{(i)})^2 = \sum_{i=1}^{N} (y^{(i)} - (w \cdot (x^{(i)} - \bar{x}) + b'))^2 \tag{2.44}$$

where we have taken into account from Eq. 2.41 that

$$\hat{y}^{(i)} = w \cdot x^{(i)} - w \cdot \bar{x} + w \cdot \bar{x} + b = w(x^{(i)} - \bar{x}) + b' \tag{2.45}$$

with $b' = b + w \cdot \bar{x}$. We introduce b' in order to simplify the mathematical calculations.

Taking the square of the residuals also has the benefit, that residuals with higher values will be counted more in the RSS, as the squared function increases faster for higher values.

The method which estimates the parameters in a regression model by minimizing the sum of the squared residuals is called ***ordinary least squares*** (OLS). This method draws a line through the data points that minimizes the sum of the squared differences between the observed values and the corresponding fitted values.

The minimum (or maximum) of a function is where its first derivative is zero. RSS is a quadratic function, thus it has only one (global) minimum. Because RSS is a function of two variables (w and b) all of its partial derivatives shall be equal to zero.

$$\begin{cases} \dfrac{\partial \text{RSS}}{\partial w} = 2\sum_{i=1}^{N}(y^{(i)} - (w(x^{(i)} - \bar{x}) + b'))(-1)(x^{(i)} - \bar{x}) \overset{set}{=} 0 \\ \dfrac{\partial \text{RSS}}{\partial b'} = 2\sum_{i=1}^{N}(y^{(i)} - (w(x^{(i)} - \bar{x}) + b'))(-1) \overset{set}{=} 0 \end{cases} \tag{2.46}$$

Dividing both sides of the second equality from Eq. 2.46 with $2N$ we can derive:

$$b' = \frac{1}{N}\sum_{i=1}^{N} y^{(i)} - w\left(\frac{1}{N}\sum_{i=1}^{N} x^{(i)} - \frac{1}{N}\sum_{i=1}^{N}\bar{x}\right) = \bar{y} - w(\bar{x} - \bar{x}) = \bar{y} \tag{2.47}$$

from where we obtain the constant term of the sample regression line:

$$b = b' - w \cdot \bar{x} = \bar{y} - w \cdot \bar{x} \tag{2.48}$$

Note

From Eq. 2.48 we can also conclude that the estimated regression line will pass through the $\{\bar{x}, \bar{y}\}$ pair, the mean of input and output.

Substituting Eq. 2.47 into the first equality of Eq. 2.46 we obtain:

$$\begin{aligned} &\sum_{i=1}^{N}(y^{(i)} - w(x^{(i)} - \bar{x}) - \bar{y}))(x^{(i)} - \bar{x}) = 0 \quad \Rightarrow \\ &\sum_{i=1}^{N}(y^{(i)} - \bar{y})(x^{(i)} - \bar{x}) - w\sum_{i=1}^{N}(x^{(i)} - \bar{x})(x^{(i)} - \bar{x}) = 0 \end{aligned} \tag{2.49}$$

from where we get the slope of the simple regression line[18]:

$$w = \frac{\sum_{i=1}^{N}(y^{(i)} - \bar{y})(x^{(i)} - \bar{x})}{\sum_{i=1}^{N}(x^{(i)} - \bar{x})^2} = \frac{S_{xy}}{S_x^2} \tag{2.50}$$

Because S_{yx} is an unbiased estimate of the population covariance, and S_x^2 is an unbiased estimate of population input variance we can also write:

$$\mathbb{E}[w] = \frac{\mathbb{C}[x, y]}{\mathbb{V}[x]} \tag{2.51}$$

Let's, write the slope in a different form by multiplying the terms in the numerator of Eq. 2.50:

$$w = \frac{\sum_{i=1}^{N}(x^{(i)} - \bar{x})y^{(i)} - \bar{y}\sum_{i=1}^{N}(x^{(i)} - \bar{x})}{\sum_{i=1}^{N}(x^{(i)} - \bar{x})^2} = \frac{\sum_{i=1}^{N}(x^{(i)} - \bar{x})y^{(i)}}{\sum_{i=1}^{N}(x^{(i)} - \bar{x})^2} \tag{2.52}$$

where we have taken into account that

[18] For a geometric interpretation of the formula check the short video from Jazon Jiao at https://youtu.be/3g-e2aiRfbU.

$$\sum_{i=1}^{N}(x^{(i)}-\bar{x})=\frac{N}{N}\sum_{i=1}^{N}x^{(i)}-\sum_{i=1}^{N}\bar{x}=N\bar{x}-\bar{x}N=0 \tag{2.53}$$

Finally, we can also check that we really find out the global minimum (and not the maximum) of the RSS. For this purpose, let's take the second derivative (derivative of the first derivative in Eq. 2.46) and check if they are positive.

$$\begin{cases}\dfrac{\partial^2 \text{RSS}}{\partial w^2}=2\sum_{i=1}^{N}(x^{(i)}-\bar{x})^2>0\\ \dfrac{\partial^2 \text{RSS}}{\partial b'^2}=2N>0\end{cases} \tag{2.54}$$

And indeed, with the assumption that there is at least one $x^{(i)}\neq\bar{x}$ and $N\neq 0$, the second derivatives are positive for all b and w so we found the global minimum.

Now, that we found out the model parameters, let's, check if they are unbiased estimate of the population attributes β_1,β_0. With the assumption that $x^{(i)}$ is not random and using equalities from Eq. 2.52 and Eq. 2.53 we obtain:

$$\begin{aligned}\mathbb{E}[w]&=\frac{\sum_{i=1}^{N}(x^{(i)}-\bar{x})\mathbb{E}\left[y^{(i)}\right]}{\sum_{i=1}^{N}(x^{(i)}-\bar{x})^2}=\frac{\sum_{i=1}^{N}(x^{(i)}-\bar{x})(\beta_1 x^{(i)}+\beta_0)}{\sum_{i=1}^{N}(x^{(i)}-\bar{x})^2}=\\&=\frac{\sum_{i=1}^{N}(x^{(i)}-\bar{x})}{\sum_{i=1}^{N}(x^{(i)}-\bar{x})^2}\beta_0+\frac{\sum_{i=1}^{N}(x^{(i)}-\bar{x})x^{(i)}}{\sum_{i=1}^{N}(x^{(i)}-\bar{x})^2}\beta_1=\beta_1\end{aligned} \tag{2.55}$$

taking into the consideration that the fraction next to β_1 is 1. If that is not obvious let's check whether the denominator is equal to the nominator:

$$\begin{aligned}\sum_{i=1}^{N}(x^{(i)}-\bar{x})^2&=\sum_{i=1}^{N}((x^{(i)})^2-x^{(i)}\bar{x}-x^{(i)}\bar{x}+\bar{x}^2)=\\&=\sum_{i=1}^{N}(x^{(i)}-\bar{x})x^{(i)}-\bar{x}\sum_{i=1}^{N}(x^{(i)}-\bar{x})=\sum_{i=1}^{N}(x^{(i)}-\bar{x})x^{(i)}\end{aligned} \tag{2.56}$$

That is, w is an unbiased estimate for β_1. Similarly, we can prove that b is an unbiased estimate for β_0:

$$\begin{aligned}\mathbb{E}[b]=\mathbb{E}[\bar{y}-w\bar{x}]=\mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N}y^{(i)}\right]-\mathbb{E}[w]\bar{x}=\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}\left[y^{(i)}\right]-\mathbb{E}[w]\bar{x}=\\=\frac{1}{N}\sum_{i=1}^{N}(\beta_1 x^{(i)}+\beta_0)-\beta_1\bar{x}=\beta_1\bar{x}+\frac{1}{N}\beta_0 N-\beta_1\bar{x}=\beta_0\end{aligned} \tag{2.57}$$

So far so good. But, there is a minor problem with calculating the values of w and b: summation is present in their formula (calculating mean also involves summation)

2.13 Summary

In this chapter, the following items have been discussed:

- The attributes of a population, like the mean and variance;
- The difference between the population and a sample from it, respective ways of estimating the population attributes from a sample;
- How to fit a simple linear regression on the data (as a sample from the population) and how to interpret the model parameters, like intercept and slope;
- How to assess to the goodness of fit of a regression model and what metrics can be used for measuring it;
- The seven assumptions of the ordinary least square regression and why they are important;
- Definition of outliers, leverage, and influential points and how they can be identified;
- Model evaluation, sampling error, confidence and prediction intervals, analysis of variance, respective the pitfalls of decisions based on p-values;
- Tasting the relevance of data pre-processing and the importance of graphing.

Lab Exercises

Jupyter Notebooks for the lab exercises are available at
https://github.com/FerencFarkasPhD/Advanced-Machine-Learning.

2.1. Check whether the height is hereditary by using Galton's dataset
(a) Read Galton's historical dataset.
(b) Plot the scatter plot of the observations.
(c) Divide the parent's mid-height into intervals and calculate the average of the children's height then make a plot.
(d) Find out the "law of regression".
(e) Create a regression model using other factors than the parents' mid-height and create the appropriate plotting.

2.2. Create distribution plots
(a) Make the PMF plot of 100 coin flips.
(b) Make the PDF and CDF of a univariate normal distribution with mean zero and unit variance. Create a fancy 3D plot of a bivariate normal distribution including the conditional and marginal probabilities.
(c) Create distributions with different skewness and kurtosis.
(d) Check the central limit theorem by creating the distribution plot of the average of two 10000 random samples each sample containing the outcome of 100, respective 500 consecutive random coin flips.
(e) Create correlation plots of random variables using different correlation coefficients.
(f) Create contour plots of the Mahalanobis distance of 1, 2, 3 standard deviations

from the mean for a bivariate normal distribution using different correlation coefficients between the random variables.
(g) Plot the likelihood function of N consecutive coin flips with k "heads" as the outcome. Use different sample size N and number of "heads" as the outcome and make your conclusions.

2.3. Predicting systolic blood pressure of a person based on age
(a) Read the observations from the `BloodPressure1` dataset introduced in Sect. 2.3 into input and output vectors.
(b) Plot the scatter plot of the observations.
(c) Find out the parameters for the simple linear regression model and plot the simple regression line.
(d) Take some input values (age) not contained in the dataset and predict the output (blood pressure).
(e) Assess the goodness of your model with regression standard error and coefficient of determination.

2.4. Predicting weight of adolescents based on their height
(a) Read the observations from the `HeightWeightSample` database discussed in Sect. 2.6.4 into input and output vectors.
(b) Plot the scatter plot of the observations.
(c) Find out the parameters for the simple linear regression model and plot the sample regression line to see its intercept.
(d) Do the same as in (c) but without using the intercept term ($b = 0$). Compare the results of the two regression models and draw your conclusions.

2.5. Checking the OLS assumptions
(a) Read the datasets discussed so far.
(b) Find out the parameters for the simple linear regression model.
(c) Create the scatter plot for the observations, respective for the residuals. Check the residual plot visually.
(d) Check the consequences of fitting a linear model when some of the OLS assumptions are broken.

2.6. Check the effect of the outliers
(a) Read the observations from the `BloodPressure1` and `Star cluster` datasets.
(b) Plot the scatter plot of the observations to check the outlier.
(c) Create a new sample by leaving out the outlier.
(d) Find out the parameters of the simple linear regression model for both samples and plot the sample regression lines.
(e) Assess the model with regression standard error and coefficient of determination and make a conclusion based on them.

2.7. Analyze the effect of sampling error
(a) Read the full `HeightWeight` dataset and assume that it represents the population of all the adolescents from a town.

(b) Plot the scatter plot of the population together with the population regression line and compare with the sample regression line.
(c) Take 10000 samples each containing 200 randomly selected adolescents from the population, and show on a single plot both the population regression line and all the estimated regression lines. Make another similar plot with only 20 observations in each sample and make your conclusion.
(d) Build up confidence intervals for the parameter estimations of the population.
(e) Make statistical tests for a single sample and make your conclusions.
(f) Make a Monte Carlo simulation using the one-sample t-test to have a deeper understanding about the p-values.

2.8. Data preprocessing
(a) Read the observations from the `TransistorCount` dataset into input and output vectors.
(b) Plot the scatter plot of the observations to check the trend.
(c) Using your insight into the data transform the data and make the scatter plot.
(d) Find out the parameters of the simple linear regression model for the transformed dataset.
(e) Assess the model with regression standard error and coefficient of determination.
(f) Check whether Moore's law supported by data.

2.9. Analyze the importance of graphing
(a) Read the observations from the `Anscombe's quartet` dataset into input and output vectors.
(b) Create a summary of the statistics to check whether they are similar (i.e., comes from the same distribution)
(c) Make the scatter plot of the observations to check the trend in each data.
(d) Make your conclusion.

References

1. ***, Durbin-Watson Significance Tables.
https://www3.nd.edu/~wevans1/econ30331/Durbin_Watson_tables.pdf
2. ***, Partition of sums of squares, Wikipedia, 2018.
https://en.wikipedia.org/wiki/Partition_of_sums_of_squares
3. ***, Pearson correlation coefficient - In least squares regression analysis, Wikipedia, 2018.
https://en.wikipedia.org/wiki/Pearson_correlation_coefficient
4. Bonate, P. L., Pharmacokinetic-Pharmacodynamic Modeling and Simulation, Springer, 2011
5. Chow, M., et al., Applied Statistics, STAT 500 online course, Department of Statistics, The Pennsylvania State University, 2018.
https://onlinecourses.science.psu.edu/stat500/
6. Chuang, W. I., Financial Econometrics, Lecture notes, Tunghai University, 2017.
http://web.thu.edu.tw/wichuang/www/
7. Field, A., Effect sizes: Null Hypothesis Significance Testing, Discovering Statistics, 2005.
http://www.discoveringstatistics.com/docs/effectsizes.pdf
8. Forsberg, L., Violation of OLS Assumption - Autocorrelation, Lecture notes, Uppsala University, Department of Statistics, 2015.

https://studentportalen.uu.se/uusp-webapp/auth/webwork/filearea/download.action?nodeId=1473435&toolAttachmentId=300125&uusp.userId=guest
9. Frost, J., Statistics By Jim - Making statistics intuitive, 2018. http://statisticsbyjim.com/
10. Galton, F., Regression Towards Mediocrity in Hereditary Stature, The Journal of the Anthropological Institute of Great Britain and Ireland, Vol. 15, 1886. http://www.stat.ucla.edu/~nchristo/statistics100C/history_regression.pdf
11. Gareth, J., et al., An Introduction to Statistical Learning with Applications in R, 8th Ed., Springer, 2018. http://www-bcf.usc.edu/~gareth/ISL/
12. Greenland, S., et. al., Statistical Tests, P-values, Confidence Intervals, and Power: A Guide to Misinterpretations, 2016, The American Statistician, Online Supplement to The ASA's Statement on p-Values: Context, Process, and Purpose http://amstat.tandfonline.com/doi/suppl/10.1080/00031305.2016.1154108
13. Hunter, J. D., Matplotlib: A 2D graphics environment, Computing In Science & Engineering, vol. 9, nr. 3, pp. 90–950, 2007, COMPUTER SOC doi: 10.1109/MCSE.2007.55. https://ieeexplore.ieee.org/document/4160265
14. Irizarry, R. A., Introduction to Data Science - Data Analysis and Prediction Algorithms with R, 2019. https://rafalab.github.io/dsbook/
15. Jones E, et al., SciPy: Open Source Scientific Tools for Python, 2001. http://www.scipy.org/
16. Lakens D., Improving your statistical inferences, Eindhoven University of Technology, Coursera, 2018. https://www.coursera.org/learn/statistical-inferences?
17. Lyon A, Why are Normal Distributions Normal?, The British Journal for the Philosophy of Science, 2014. http://aidanlyon.com/aidanlyon.com/media/publications/Lyon-normal_distributions.pdf
18. Kutner, M. H., et al., Applied Linear Statistical Models, 5th Ed., McGraw-Hill/Irwin, 2005. https://mysite.science.uottawa.ca/rkulik/mat3378/mat3378-textbook.pdf
19. Neter, J., et al., Applied Linear Regression Models, Richard D. Irwin, Inc, 1983.
20. Pedregosa F. et al., Scikit-learn: Machine Learning in Python, JMLR 12, pp. 2825-2830, 2011. http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html
21. Pérez, F., Granger, B. E., IPython: A System for Interactive Scientific Computing, Computing in Science and Engineering, vol. 9, no. 3, pp. 21-29, May-June 2007 doi: 10.1109/MCSE.2007.53 https://ipython.org
22. Sainani, K., Statistics in Medicine, Online Course, Stanford University. https://lagunita.stanford.edu/courses/Medicine/MedStats-SP/SelfPaced/about
23. Simon, L., et al., Probability Theory and Mathematical Statistics, STAT 414/415 online course, Department of Statistics, The Pennsylvania State University, 2018. https://onlinecourses.science.psu.edu/stat414/
24. Simon, L., et al., Applied Regression Analysis, STAT 462 online course, Department of Statistics, The Pennsylvania State University, 2018. https://online.stat.psu.edu/stat462/
25. Simon, L., et al., Regression Methods, STAT 501 online course, Department of Statistics, The Pennsylvania State University, 2018. https://onlinecourses.science.psu.edu/stat501/
26. Taboga, M., Gauss Markov theorem, StatLect, 2017. https://www.statlect.com/fundamentals-of-statistics/Gauss-Markov-theorem
27. Trefethen, L. N., et al., Numerical Linear Algebra, Society for Industrial and Applied Mathematics, 1997.
28. Ullah, M. I, Consequences of Heteroscedasticity for OLS, Basic Statistics and Data Analysis, Lecture notes, 2013. http://itfeature.com/heteroscedasticity/consequences-of-heteroscedasticity-for-ols

29. Wasserstein, R., ASA News, American Statistical Association Releases Statement on Statistical Significance and p-values, 2016.
http://www.amstat.org/asa/files/pdfs/P-ValueStatement.pdf
30. Wasserstein, R. L., et al., The ASA's Statement on p-Values: Context, Process, and Purpose, The American Statistician, 70:2, 129-133, 2016., DOI: 10.1080/00031305.2016.1154108
https://doi.org/10.1080/00031305.2016.1154108
31. Willmott, C. J., et al., Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance, Inter-Research, Vol. 30: 79–82, 2005.
https://www.int-res.com/articles/cr2005/30/c030p079.pdf

Chapter 3
Multiple Linear Regression

3.1 Introduction

Throughout Chapter 2, we have seen the concept of simple linear regression where a single predictor variable x was used to model the response variable y. In many applications, however, there is more than one factor that influences the response y. ***Multiple regression models*** thus describe how a single response variable y depends linearly on several predictor variables x. Regression models having only one response variable y (whether there are only one or multiple independent variables) are also called ***univariate regression models***. There are also regression models with two or more response variables y, and these models are usually called ***multivariate regression models***. In this chapter, we will focus mainly to discuss the multiple linear regression (one output variable and several input variables), and only in the last paragraph, we will briefly mention the multivariate regression model (multiple inputs and multiple output variables).

3.2 Simple Dataset with Two Explanatory Variables

Again, let's start our analysis with a simple dataset. In Table 2.1 we had the result of a survey, containing only the age of a person together with the measured systolic blood pressure. However, already at that time was stated that not only the age but other factors like weight, illness, side effects of the drug, should also contribute to the blood pressure of a person.

Let's, imagine for the sake of the game that another survey is conducted with a very limited budget that will contain not only the age but the measured weight and blood pressure of the people. The result of this very small survey is shown in Table 3.1 with $N = 11$ observations. There are two input variables, `Age` and `Weight`, denoted by x_1 and x_2, and one output variable `Blood pressure` denoted by y. For indexing the observations, again, we can use a superscript index as shown in Table 3.1. You might ask why such a small dataset used again? The advantage can be seen in the lab exercise when you can "see and feel" the data manipulated.

Table 3.1 Systolic blood pressure as function of age and weight (`BloodPressure2` dataset)

$\mathbf{x}_1$	Age [year]	$\mathbf{x}_2$	Weight [pounds]	$\mathbf{y}$	Systolic blood pressure
$x_1^{(1)}$	52	$x_2^{(1)}$	173	$y^{(1)}$	132
$x_1^{(2)}$	59	$x_2^{(2)}$	184	$y^{(2)}$	143
$x_1^{(3)}$	67	$x_2^{(3)}$	194	$y^{(3)}$	153
$x_1^{(4)}$	73	$x_2^{(4)}$	211	$y^{(4)}$	162
$x_1^{(5)}$	64	$x_2^{(5)}$	196	$y^{(5)}$	154
.....	.....	.....	.....	.....	.....
$x_1^{(N)}$	72	$x_2^{(N)}$	217	$y^{(N)}$	166

Data source: http://college.cengage.com/mathematics/brase/understandable_statistics/7e/students/datasets/mlr/frames/mlr02.html

This dataset is still denoted with $\mathcal{D}_N = \{\{\mathbf{x}^{(1)}, y^{(1)}\}, \{\mathbf{x}^{(2)}, y^{(2)}\}, ..., \{\mathbf{x}^{(N)}, y^{(N)}\}\}$. Please, note that now the input observations are denoted as vectors in contrast to Sec. 2.3.

Because there are two input variables in this case, instead of having an estimated regression line, an estimated regression plane will be obtained as shown in Fig. 3.1 where some data points are "above" the estimated regression plane and others are "below" the estimated regression plane. Thus, some of the residuals will be positive and others will be negative in a similar way as we had with the estimated regression line with one input variable. There should be no surprise that the systolic blood pressure increases not only with the age but also with the weight of a person.

Of course, there could be other factors influencing the blood pressure, so further input variables might be included if another survey is conducted. For the general case, the number of predictor variables is denoted by D ($D = 2$ for this simple example) and referred to as the dimension of the input space. Then the result of the regression analysis will be a D dimensional regression hyperplane. A multiple linear regression model with D predictor variables $x_1, x_2, ..., x_D$ and a response y, can be written as:

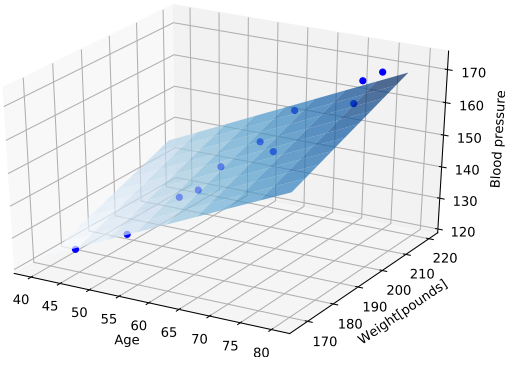


Fig. 3.1 Systolic blood pressure as the function of age and weight estimated with a linear regression plane (`BloodPressure2` dataset)

$$\mu_{y|(x_1,x_2,\ldots,x_D)} = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \ldots + \beta_D \cdot x_D$$
$$y^{(i)} = \beta_0 + \beta_1 \cdot x_1^{(i)} + \beta_2 \cdot x_2^{(i)} + \ldots + \beta_D \cdot x_D^{(i)} + \varepsilon^{(i)} \quad \text{for} \quad i = 1,2,\ldots \tag{3.1}$$

As in the case of simple linear regression, ε is a random variable with zero mean and constant variance, representing the error between the conditional mean $\mu_{y|(x_1,x_2,\ldots,x_D)}$ and the actual value of the member from the population. However, the conditional mean of y is not a line, as in the case of the simple linear regression, but a (hyper)plane. Thus, correspondingly, we can write for the estimated hyperplane:

$$\hat{y} = b + w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_D \cdot x_D$$
$$y^{(i)} = b + w_1 \cdot x_1^{(i)} + w_2 \cdot x_2^{(i)} + \ldots + w_D \cdot x_D^{(i)} + r^{(i)} \quad \text{for} \quad i = \overline{1,N} \tag{3.2}$$

where N denotes the number of observations in the sample.

Note

The `BloodPressure2` *dataset shows high collinearity among the predictors. This will be discussed in more detail in Sec. 3.6.1.*

3.3 OLS for Multiple Linear Regressions

Our goal with ordinary least squares regression is to fit a hyperplane into $(D+1)$-dimensional space that minimizes the sum of squared residuals (RSS).

3.3.1 Analytical Solution

Again, we can write the residual sum of squares:

$$\text{RSS} = \sum_{i=1}^{N} (r^{(i)})^2 = \sum_{i=1}^{N} \left(y^{(i)} - b - \sum_{j=1}^{D} w_j \cdot x_j^{(i)} \right)^2 \tag{3.3}$$

using the second equality from Eq. 3.2. As in the case of simple linear regression, we can use OLS to find the minimum of the residual sum of squares. That is, we take the first derivatives with respect to the model parameters b, w_1, w_2, ..., w_D and set them equal to zero.

$$\begin{cases} \dfrac{\partial \text{RSS}}{\partial b} = (-2) \displaystyle\sum_{i=1}^{N} r^{(i)} \overset{set}{=} 0 \\ \dfrac{\partial \text{RSS}}{\partial w_1} = (-2) \displaystyle\sum_{i=1}^{N} r^{(i)} x_1^{(i)} \overset{set}{=} 0 \\ \quad \ldots \qquad\qquad \ldots \\ \dfrac{\partial \text{RSS}}{\partial w_D} = (-2) \displaystyle\sum_{i=1}^{N} r^{(i)} x_D^{(i)} \overset{set}{=} 0 \end{cases} \tag{3.4}$$

Please, note that the residuals are constrained by $D+1$ equations. Thus, the number of degrees of freedom for the residuals with N observations and D number of independent variables is only $N-D-1$.

While it is possible to estimate the parameters of multiple linear models with a similar method that we used in Chapter 2, the computations become very complicated quickly. Therefore, we will employ linear algebra methods using vectors and matrices to make the computations more efficient and our life easier.

With the introduction of the notation $b \equiv w_0$ we can create the following vectors of weight, output, and residuals:

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_D \end{bmatrix}; \quad \mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix}; \quad \mathbf{r} = \begin{bmatrix} r^{(1)} \\ r^{(2)} \\ \vdots \\ r^{(N)} \end{bmatrix}; \tag{3.5}$$

Please, note that the indexes (and the dimensions) of the two vectors $\mathbf{w}$ and $\mathbf{y}$ are different. Vector $\mathbf{w}$ has dimension $D+1$ corresponding to the number of input variables plus the constant term $b \equiv w_0$, and the indexes correspond to the input variables. Vector $\mathbf{y}$ has dimension N corresponding to the number of observations in the sample, and the indexes identify the observations (number of raw in Table 3.1). From this table, we can also observe that the input values are organized in multiple columns, each column representing one input variable with N observations. Thus, the input values can be written - considering the general case with D number of input variables - in the following matrix form:

$$\mathbf{X} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_D^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_D^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(N)} & x_2^{(N)} & \dots & x_D^{(N)} \end{bmatrix}; \quad \dot{\mathbf{X}} = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} & x_2^{(1)} & \dots & x_D^{(1)} \\ x_0^{(2)} & x_1^{(2)} & x_2^{(2)} & \dots & x_D^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_0^{(N)} & x_1^{(N)} & x_2^{(N)} & \dots & x_D^{(N)} \end{bmatrix} \tag{3.6}$$

The first column in the second matrix denoted with $\dot{\mathbf{X}}$ represents the dummy input $x_0^{(i)} = 1$ ($i = \overline{1,N}$) corresponding to the intercept term $b \equiv w_0$. This extended input matrix is called the ***design matrix***, also known as ***model matrix*** or ***regressor matrix***. The introduction of this matrix will help us writing the OLS solution in compact form.

The design matrix is organized into rows and columns as you are already familiar with. Each column represents the N observations for a specific input variable, e.g. the observations of the j-th variable is denoted by $\mathbf{x}_j^\top = \begin{bmatrix} x_j^{(1)} & x_j^{(2)} & \dots & x_j^{(N)} \end{bmatrix}$ for $j = \overline{0,D}$. Likewise, each row represents one $D+1$ dimensional observation for the input $x_0, x_1, x_2, ..., x_D$, e.g. the i-th observation is denoted by $\dot{\mathbf{x}}^{(\mathbf{i})} = \begin{bmatrix} x_0^{(i)} & x_1^{(i)} & x_2^{(i)} & \dots & x_D^{(i)} \end{bmatrix}^\top$

for $i = \overline{1,N}$. Thus, the design matrix can be written in the following two ways[1]:

$$\dot{\mathbf{X}} = \left[\mathbf{x}_0\ \mathbf{x}_1\ \mathbf{x}_2\ \ldots\ \mathbf{x}_D\right]; \quad \dot{\mathbf{X}} = \begin{bmatrix} \left(\dot{\mathbf{x}}^{(1)}\right)^\top \\ \left(\dot{\mathbf{x}}^{(2)}\right)^\top \\ \vdots \\ \left(\dot{\mathbf{x}}^{(N)}\right)^\top \end{bmatrix} \tag{3.7}$$

With these matrix and vector notations, we can write the second line of Eq. 3.2 in the following form:

$$\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \ldots & x_D^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \ldots & x_D^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & x_2^{(N)} & \ldots & x_D^{(N)} \end{bmatrix} \cdot \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_D \end{bmatrix} + \begin{bmatrix} r^{(1)} \\ r^{(2)} \\ \vdots \\ r^{(N)} \end{bmatrix} \quad \Leftrightarrow \quad \mathbf{y} = \dot{\mathbf{X}} \cdot \mathbf{w} + \mathbf{r} \tag{3.8}$$

where matrix dot product used between matrix $\dot{\mathbf{X}}$ and vector $\mathbf{w}$. The residual sum of squares can be also written in compact matrix notations using Eq. 3.8:

$$\text{RSS} = \sum_{i=1}^{N} (r^{(i)})^2 = \mathbf{r}^\top \cdot \mathbf{r} = (\mathbf{y} - \dot{\mathbf{X}} \cdot \mathbf{w})^\top \cdot (\mathbf{y} - \dot{\mathbf{X}} \cdot \mathbf{w}) \tag{3.9}$$

The constraints in Eq. 3.4 can be also written in compact matrix form:

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1^{(1)} & x_1^{(2)} & \ldots & x_1^{(N)} \\ x_2^{(1)} & x_2^{(2)} & \ldots & x_2^{(N)} \\ \vdots & \vdots & \ddots & \vdots \\ x_D^{(1)} & x_D^{(2)} & \ldots & x_D^{(N)} \end{bmatrix} \cdot \begin{bmatrix} r^{(1)} \\ r^{(2)} \\ \vdots \\ r^{(N)} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \Leftrightarrow \quad \dot{\mathbf{X}}^\top \cdot \mathbf{r} = \mathbf{0} \tag{3.10}$$

from where, substituting the vectorized form of the residual from Eq. 3.8, we obtain:

$$\dot{\mathbf{X}}^\top \cdot (\mathbf{y} - \dot{\mathbf{X}} \cdot \mathbf{w}) = 0 \quad \Rightarrow \quad \dot{\mathbf{X}}^\top \cdot \mathbf{y} = \dot{\mathbf{X}}^\top \cdot \dot{\mathbf{X}} \cdot \mathbf{w} \tag{3.11}$$

Thus, the analytical solution for the model parameters is obtained as[2]:

$$\mathbf{w} = (\dot{\mathbf{X}}^\top \dot{\mathbf{X}})^{-1} \dot{\mathbf{X}}^\top \mathbf{y} \tag{3.12}$$

[1] Dot is placed on a vector or a matrix symbol whenever the dummy variable $x_0 = 1$ corresponding to the intercept term is included.

[2] Matrix dot operators will be only explicitly used to separate variable names. On the other hand, a matrix element-wise operator will be shown always. Thus, if no operator is present between matrices or vectors, the matrix dot product shall be understood.

where we assumed that the matrix inverse of the product $\dot{\mathbf{X}}^\top\dot{\mathbf{X}}$ exists. If the inverse does not exist, the normal equations can still be solved, but the solution may not be unique. The inverse of $\dot{\mathbf{X}}^\top\dot{\mathbf{X}}$ exists, if the columns of $\dot{\mathbf{X}}$ are linearly independent (no collinearity between the predictor variables - see assumption in Sec. 2.7.6). That is, no column of the design matrix can be written as a linear combination of the other columns. Based on Eq. 3.12 we can write the vector of estimated values as:

$$\hat{\mathbf{y}} = \dot{\mathbf{X}}\cdot\mathbf{w} = \dot{\mathbf{X}}(\dot{\mathbf{X}}^\top\dot{\mathbf{X}})^{-1}\dot{\mathbf{X}}^\top\mathbf{y} = \mathbf{H}\cdot\mathbf{y} \tag{3.13}$$

The $N \times N$ matrix $\mathbf{H} = \dot{\mathbf{X}}(\dot{\mathbf{X}}^\top\dot{\mathbf{X}})^{-1}\dot{\mathbf{X}}^\top$ is known as the ***hat-matrix***, sometimes also called the ***influence matrix*** or ***projection matrix***. It maps the vector of response values $\mathbf{y}$ to the vector of fitted values $\hat{\mathbf{y}}$ that lie on the regression hyperplane (see Fig. 3.2). Thus, the projection matrix naming is straight forward. The name of hat matrix comes from the fact that $\mathbf{H}$ "makes" from output $\mathbf{y}$ a hat-output $\hat{\mathbf{y}}$.

Now, that we have an intuition about why $\mathbf{H}$ is called hat-matrix or projection matrix, let's see why it is called the influence matrix. We can write Eq. 3.13 in a verbose form showing only elements that are required to calculate $\hat{y}^{(i)}$:

$$\begin{bmatrix}\dots\\ \dots\\ \dots\\ \hat{y}^{(i)}\\ \dots\\ \dots\\ \dots\end{bmatrix} = \begin{bmatrix}\ddots & & & & & & \\ & \ddots & & & & & \\ & & \ddots & & & & \\ h_{i1} & h_{i2} & \dots & \boxed{h_{ii}} & \dots & \dots & h_{iN} \\ & & & & \ddots & & \\ & & & & & \ddots & \\ & & & & & & \ddots\end{bmatrix} \cdot \begin{bmatrix}y^{(1)}\\ y^{(2)}\\ \dots\\ \boxed{y^{(i)}}\\ \dots\\ \dots\\ y^{(N)}\end{bmatrix} \tag{3.14}$$

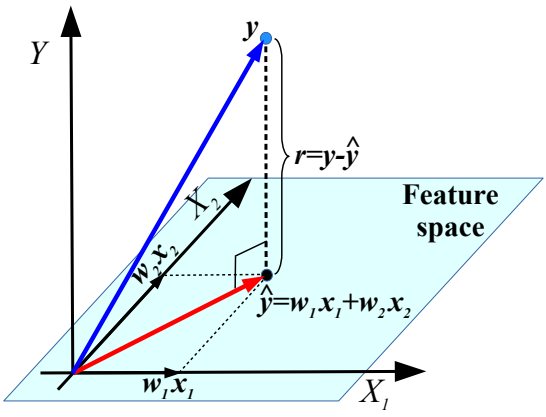


Fig. 3.2 Projection of the output variable on the (hyper)plane of the input variables (omitting the constant term does not affect the generalization, as the intercept only shifts the (hyper)plane to a parallel one)

where again β_0 and β_1 are the population attributes to be estimated. Taking logarithms on both sides of the power curve equation gives:

$$\log \hat{y} = \underbrace{\log \hat{\beta}_0}_{=w_0} + w_1 \log x \quad \Rightarrow \quad y' = w_0 + w_1 x' \tag{3.213}$$

Thus an equivalent way to write a power curve equation is that the logarithm of y is a straight-line function of the logarithm of x. This regression equation is sometimes referred to as a ***log-log regression***.

How shall we interpret the regression coefficients in the case of log-log regression? The answer is this: interpret the regression coefficient as the percent increase (or decrease) in the dependent variable for every 1% increase in the independent variable. However, this statement is approximately true only if the dependent variable changes not more than $\pm 10\%$ with a 1% change of the independent variable. The interpretation of the regression coefficients can be deduced as follows:

$$\hat{y}' - \hat{y}'_o = \log \frac{\hat{y}}{\hat{y}_o} = w_0 - w_0 + w_1(\log x - \log x_o) = w_1 \log \frac{x}{x_o} \tag{3.214}$$

3.5.6.2 Case Study: Mammal Species

Let's see another case study where want to fit a multiple linear regression model on the `Mammal species` dataset with response variable `Y` as the Brain weight (in gram), respective predictors `Xb` as Body weight (in kilograms), `Xg` as Gestation period length (in days), and `Xl` as Litter size. To have some idea about the relationship between predictors and response variables, let's create the scatter plot of a pair of responses and each predictor (see the SRL results on Fig. 3.29). It should be observed on the scatter plots that all three exhibit at least one outlier. The correlation coefficient between brain and body, respective brain, and gestation is high.

Let's create the matrix of histograms for all four variables, as shown in Fig. 3.30. It can be observed that all four variables have a distribution skewed to the left. Remember that statistical inference related to standard errors and confidence intervals

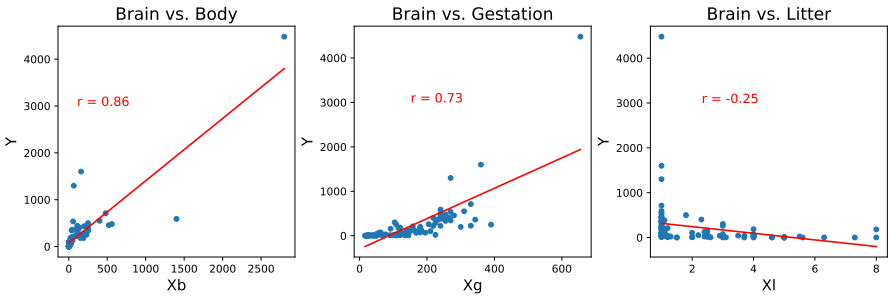


Fig. 3.29 Simple linear regression of the `Mammal species` dataset using each predictor individually (Pearson correlation coefficient between each predictor and response is also presented)

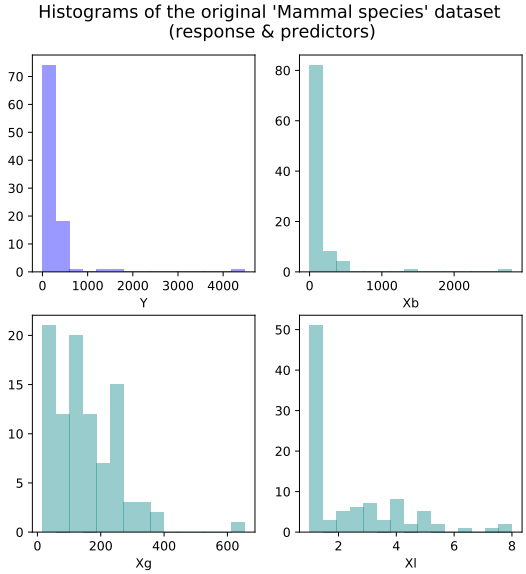


Fig. 3.30 Histogram of the original `Mammal species` dataset for both input (green) and output (blue) variables (all four variables have a distribution skewed to the left)

assume that the errors have a normal distribution. But due to suspected outliers and skewed distribution of the data, this assumption is not satisfied, as can be seen on the left of Fig. 3.15 where the studentized residual Q-Q normal probability plot is shown for the multiple linear regression. Moreover, according to the Goldfeld-Quandt test, the null hypothesis of the error having constant variance can be rejected. Thus, the OLS assumption of the constant variance of the error is also violated.

Table 3.17 MLR results for the original `Mammal species` dataset using all observations

Input	Coefficient	Std. error	t-statistic	p-value	LCL	UCL
Intercept	-225.2921	83.059	-2.712	0.008	-390.254	-60.330
Xb	0.9859	0.094	10.457	0.000	0.799	1.173
Xg	1.8087	0.354	5.103	0.000	1.105	2.513
Xl	27.6486	17.414	1.588	0.116	-6.938	62.235
$\hat{\sigma}$=224.56	adj-R^2=0.80				$N_{\text{obs}} = 96$	

While the adjusted R-squared value is relatively high, that is, 80% of the mammal's brain variation is explained by the body weight, gestation period, and litter size, the RMSE=224.56 grams is quite high either. Thus, besides the non-constant variance of the error and the non-normal error distribution problem, the high RMSE value makes our model useless if you take into account that the median of the response variable is 75 grams. That is a clear example of why you should not rely only on the R-squared value and use your common sense and domain knowledge.

Let's search for outliers (influential observations) using the leave-one-out methods. There are several observations marked as problematic but looking at the squared

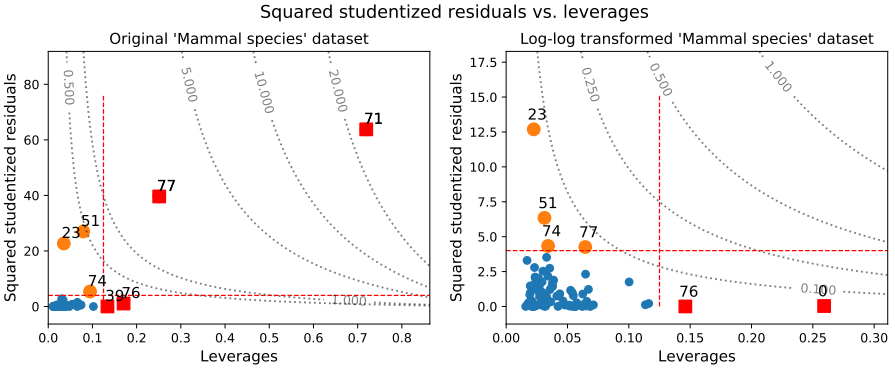


Fig. 3.31 Squared studentized residual vs. leverages together with the contour of Cook's distances based on the MLR on the original `Mammal species` dataset on the left, respective on the log-log transformed `Mammal species` dataset on the right

studentized residuals vs. leverage plot shown on the left of Fig. 3.31, it is clear that there are at least two influential observations indexed as 71 and 77 that have Cook's distance well above the threshold of 1.

However, removing these two observations from the dataset, the R-squared value drops to 0.53, and the RMSE, although dropped, is still 172.8 grams (see Table 3.18). We still have a problem with heteroskedasticity and non-normality of the residuals, too. Moreover, according to t-statistic, we cannot reject the null hypothesis that the regression coefficient of `Xl` is significantly different from zero. Removing this feature from the MLR does not solve the problem either. So using the MLR, we end up at a dead end.

Table 3.18 MLR results for the original `Mammal species` dataset w. removed observations

Input	Coefficient	Std. error	t-statistic	p-value	LCL	UCL
Intercept	-138.8112	66.771	-2.079	0.040	-271.463	-6.159
Xb	0.4776	0.202	2.366	0.020	0.077	0.879
Xg	1.6885	0.309	5.472	0.000	1.075	2.302
Xl	13.7929	13.752	1.003	0.319	-13.528	41.114
$\hat{\sigma}$=172.8	adj-R^2=0.53				$N_{\text{obs}} = 94$	

Let's take the logarithm of both the independent and dependent variables, and try to fit the linear regression model (also called the log-log model). After the logarithm transformation of both sides, the distributions of the variables will become less skewed, as can be seen in Fig. 3.32.

Let's create the scatter plot of the pair of responses and each predictor again, but at this time, using the log-transformed data. It is visible from Fig. 3.33 that a linear relationship is present, and the Pearson correlation coefficients between the response and predictors are much higher for all three predictors. That is another example of why data visualization is so essential in data science. Just looking at pure numbers,

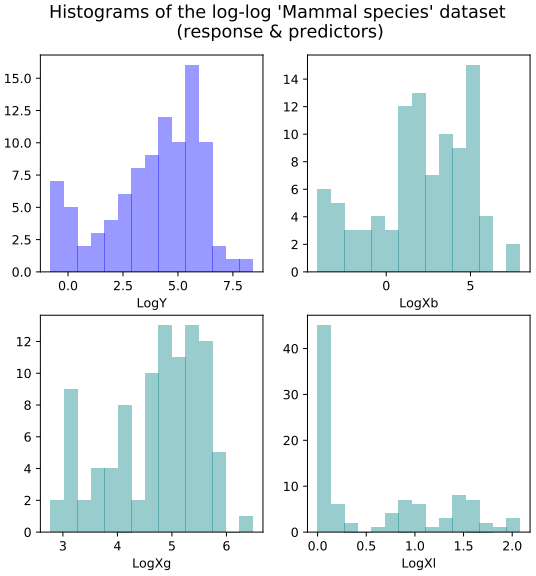


Fig. 3.32 Histogram of the log-log transformed `Mammal species` dataset for both input (green) and output (blue) variables (Now all four variables have much lesser positive skewness)

the difference between the correlation coefficients of the original data and the log-transformed data is not so remarkable for the first two predictors: for `Xb` increased from 0.86 to 0.96, and for `Xg` from 0.73 to 0.89. However, the scatter plots tell a completely different story: there are no outliers anymore.

And not only the scatter plots tell this, but the leave-one-out method also confirms this: now none of the observations has a Cook's distance above 0.1 (see the right of Fig. 3.31). Moreover, there is no heteroskedasticity issue anymore, and the residuals have an almost normal distribution (see on the right of Fig. 3.15). The R-squared value is now 0.95, and more importantly, the RMSE drops to 0.48. However, this is not in the unit of the original data, so we may exponentiate it to get $\exp(0.48) = 1.62$ grams. There is one minor problem, though: according to the t-statistic, the intercept

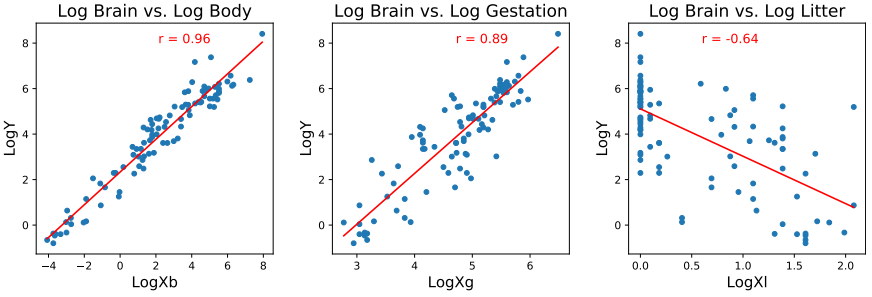


Fig. 3.33 Simple linear regression of the `Mammal species` dataset using each predictor individually (Pearson correlation coefficient between each predictor and response is also presented)

coefficient is significantly not different from zero. Fitting an MLR by omitting the intercept provides a model with an adjusted R-square value of 0.99 (see Table 3.19).

Table 3.19 MLR results for the log-log transformed `Mammal species` dataset

Input	Coefficient	Std. error	t-statistic	p-value	LCL	UCL
LogXb	0.5449	0.023	23.878	0.000	0.500	0.590
LogXg	0.5979	0.021	29.113	0.000	0.557	0.639
LogXl	-0.1990	0.078	-2.549	0.012	-0.354	-0.044
$\hat{\sigma}$=0.48	adj-R^2=0.99				$N_{\text{obs}} = 96$	

Mean and standard error of the original dependent variable

Once you have got the regression model with the log-transformed dependent variable, you may be interested in how the obtained estimated mean and standard deviation is related to the original variable. Before answering that question, we shall take a detour and see how moments of the normal respective log-normal distributions are related.

The ***moment-generating function*** of a real-valued random variable X is an alternative specification of its probability distribution and it is defined as $M_X(t) = \mathbb{E}[\exp(tX)]$ for $t \in \mathbb{R}$. The moment-generating function is so named because it can be used to find the moments of the distribution. Using the series expansion of $\exp(tX)$ we can write:

$$M_X(t) = \mathbb{E}[\exp(tX)] = 1 + t\mathbb{E}[X] + \frac{t^2\mathbb{E}[X^2]}{2!} + \ldots + \frac{t^n\mathbb{E}[X^n]}{n!} + \ldots = \\ = 1 + tm_1 + \frac{t^2 m_2}{2!} + \ldots + \frac{t^n m_n}{n!} + \ldots \tag{3.215}$$

where m_n is the n-th moment. Differentiating $M_X(t)$ i times with respect to t and setting $t = 0$, we obtain the i-th moment about the origin, m_i. If X is a continuous random variable, the moment-generating function's definition expands to[22]:

$$M_X(t) = \mathbb{E}[\exp(tX)] = \int_{-\infty}^{+\infty} \exp(tx) f_X(x) \mathrm{d}x \tag{3.216}$$

where $f_X(x)$ denotes the probability density function (PDF). For a normal distribution $\mathcal{N}(x; \mu, \sigma^2)$, the PDF is:

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\Big(-\frac{(x-\mu)^2}{2\sigma^2}\Big) \tag{3.217}$$

Then for a standard normal distribution, denoted by $\mathcal{N}(z; 0, 1)$, the corresponding moment generating function is defined by:

[22] See Wikipedia at https://en.wikipedia.org/wiki/Moment-generating_function

3.5.8.4 Case Study: Salary Discrimination

The Equal Pay Act[46] became law in 1963, and it mandates employers pay workers substantially equal pay for performing the same job regardless of their gender. Let's see how this is carried out in practice. The `SalaryDiscrimination` dataset[47] is a result of a study of faculty salaries in a small college in the Midwest conducted in the 1970s, consisting of observations on six variables for 52 tenure-track professors. It has the dependent variable `salary` (academic year salary, in US dollars), the independent numerical variables `years` (the number of years employed at this college), respective `yearDegree` (number of years since the highest degree was earned). It also has three nominal input variables: `sex` (male or female), `degree` (masters or doctorate), and `rank` (assistant, associate or full professor). The MLR results are shown in the upper part of Table 3.31.

Table 3.31 MLR results for `SalaryDiscrimination` dataset

Input	Coefficient	Std. error	t-statistic	p-value	LCL	UCL
Intercept	1.691e+04	816.442	20.715	0.000	1.53e+04	1.86e+04
rank[associate]	5292.3608	1145.398	4.621	0.000	2985.411	7599.311
rank[full]	1.112e+04	1351.772	8.225	0.000	8396.155	1.38e+04
degree[masters]	1388.6133	1018.747	1.363	0.180	-663.248	3440.475
sex[male]	-1166.3731	925.569	-1.260	0.214	-3030.565	697.818
years	476.3090	94.914	5.018	0.000	285.143	667.475
yearDegree	-124.5743	77.486	-1.608	0.115	-280.640	31.491
$\hat{\sigma}$=2398.42	R^2=0.836	(full model)				
Intercept	1.62e+04	638.677	25.370	0.000	1.49e+04	1.75e+04
rank[associate]	4262.2847	882.891	4.828	0.000	2487.113	6037.457
rank[full]	9454.5232	905.830	10.437	0.000	7633.230	1.13e+04
years	375.6956	70.918	5.298	0.000	233.106	518.285
$\hat{\sigma}$=2402.22	R^2=0.835	(reduced)				

As can be seen in the upper part of Table 3.31, the p-values associated with the coefficient estimates for the two dummy variables `sex` and `degree` are very large, suggesting no statistical evidence of a real difference in salary between genders and degree. The same applies to the years since the degree was obtained (`yearDegree`).

Because both the point-biserial correlation coefficient and phy coefficient are a special case of Pearson correlation coefficient, we can create a correlation matrix with all the variables, except the nominal variable `rank`. As a result, the dependent variable `salary` has a high correlation with both independent numerical variables: `years` (0.70) and `yearDegree` (0.67). On the other hand, there is a high correlation between the two numerical input variables (0.64). The response variable has no correlation with the binary variable `degree` (0.07) and a very low correlation with the binary variable `sex` (0.25).

[46] See at https://www.eeoc.gov/statutes/equal-pay-act-1963

[47] The dataset can be downloaded from https://data.princeton.edu/wws509/datasets/#salary.

The contingency table between the two nominal variables `sex` and `rank` is presented in Table 3.32. Using Eq. 3.284 and Eq. 3.285 we obtain the asymmetric correlations: $\lambda_{sex|rank} = -1.0$, respective $\lambda_{rank|sex} = 0.56$. The symmetric correlation can be obtained from Eq. 3.286, i.e., $\lambda = 0.09$. Similarly, we can construct the contingency table between nominal variables `degree` and `rank` to obtain the asymmetric lambda values $\lambda_{degree|rank} = -0.56$, respective $\lambda_{rank|degree} = 0.56$, and the symmetric lambda value $\lambda = 0.16$. The eta correlation ratio between the response variable `salary` and the nominal variable `rank` is 0.87.

Table 3.32 Crosstab between nominal variables from `SalaryDiscrimination` dataset

Contingency table		sex		
		female	**male**	**Total**
rank	**assistant**	$N_{11} = 8$	$N_{12} = 10$	$N_{1*} = 18$
	associate	$N_{21} = 2$	$N_{22} = 12$	$N_{2*} = 14$
	full	$N_{31} = 4$	$N_{32} = 16$	$N_{3*} = 20$
	Total	$N_{*1} = 14$	$N_{*2} = 38$	$N = 52$

In the lower part of Table 3.31, the MLR result of the reduced model is shown with only two input variables: `years` and `rank`. These are the predictors with high correlation with the response (remember, the two numerical input variables are also highly correlated, so we should drop the one with a high p-value). This reduced model offers the same goodness of fit as the full model. Thus, from the linear regression model, we cannot infer any salary discrimination between genders.

Now, using the advantage of stratification, let's see what the visual inspection might reveal about latent discrimination. Analyzing the left of Fig. 3.45, which represents a category scatter plot, it is obvious that there are fewer females than males among professors (representing only 27% from the total), and the salaries of fe-

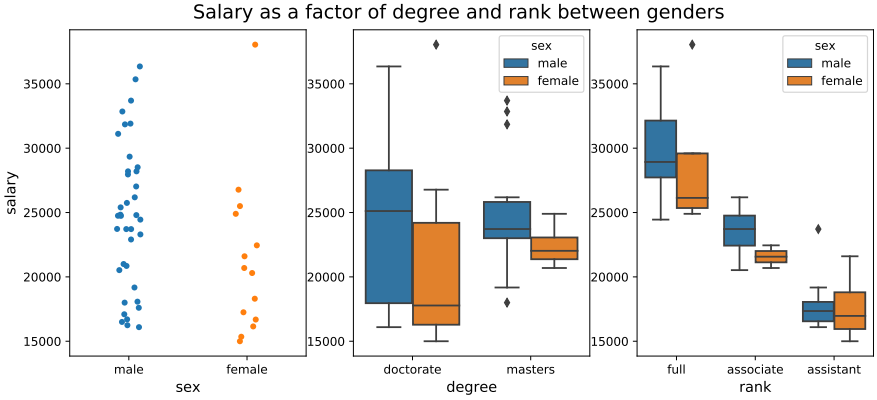


Fig. 3.45 Scatter plot of the salary as a function of sex (on the left), box plot of the salary as a factor of degree and sex (in the middle), respective the salary as a factor of rank and sex (on the right) from the `SalaryDiscrimination` dataset

males are lower compared to the males (although there is one outlier among females, which provides the absolute maximum among all the wages). The same observation we can get from the middle of Fig. 3.45, where the salaries as a factor of degree and gender are shown in a box plot format. A similar observation can be obtained from the right to Fig. 3.45, where the salaries as a factor of rank and gender are presented (though, for assistant professors, there is no such visible difference).

Now, the question is, how is possible, that statistically, no proof of salary discrimination between genders, but the stratification suggests otherwise? Since in our reduced model the `salary` is dependent only on `years` and `rank`, let's analyze these two input variables. For example, the `years` is weakly correlated with `sex` (0.38) but we can observe that $\texttt{years} \approx 4.07 + 4.66 \cdot \texttt{sex|male}$ with a very low p-value for the gender coefficient, suggesting that there is statistical evidence of a real difference between genders in the number of years since they were hired at the college. As the salary increases by 375.7 US dollars every year since an individual is hired at this college (see the lower part of Table 3.31), we already have an explanation of why in average male professors have a salary higher with $4.66 \cdot 375.7 \cong 1751$ US dollars than females. That indicates that females were less accepted being professors at college in the past. On the other hand, with the baseline of assistant professors, an associate professor receives annually 4262 US dollars in plus, while a full professor 9455 US dollars in plus. However, from Fig. 3.46 it is clear that, while the distribution of ranks with a master degree within a gender is around the same for females and males, with a doctorate this is not the case: 70% of females with a doctorate has an assistant rank, none of them is associate professor, and only 30% has full professor rank, while in case of males with doctorate only 29% are assistants, but 21% are associate professors, and 50% are full professors. Thus, there is a piece of clear evidence that in the promotion to associate or full professor, females are discriminated against negatively. And that affects the annual salaries, as well, since

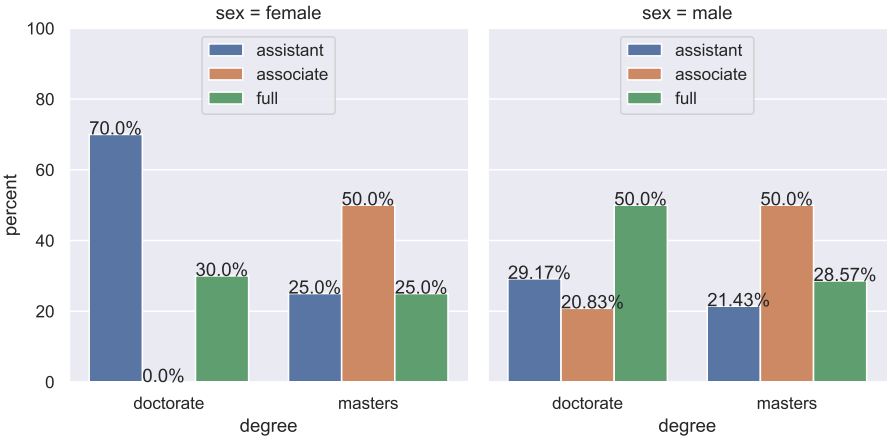


Fig. 3.46 Percentage bar plot as a distribution of ranks for each subgroup (master, respective doctorate) separately for male and female professors from the `SalaryDiscrimination` dataset

from our model, with the assistant professor as the baseline, an associate professor has on average 4262 US dollars higher annual salary, while a full professor on the average earns 9455 US dollars in plus every year. The fact that females, on average, have a lower number of years since employment, plus females do not get promoted even with doctorate degrees like males, provides an adequate explanation of why female salaries on average are lower at this college.

We conclude that, although there is no statistical evidence of salary discrimination between genders, there are latent discriminations, some inherited from the past, some were still present when the dataset was recorded. These are:

- females were still underrepresented among professors during the 1970s at this college;
- females, on average, had a lower number of years since they were hired at this college than males, which was a direct consequence of lower salary for females than men (that could be the result of a legacy of females were less accepted being professors at college in the past, as well as a legacy of discrimination in the education system);
- even during the 1970s, in the promotion to associate or full professor, females were discriminated against negatively.

3.5.8.5 Summary of Association Measures

There are several correlations and association measures introduced in the statistical literature. In this book, however, only the most widely used measures have been discussed. In Table 3.33, a summary of correlation and association measures between different types of variables are presented concisely.

Table 3.33 Correlation and association measures between different type of variables

Associations		y			
		binary	**ordinal**	**nominal**	**continuous**
x	**binary**	phi	Spearman, Cramér's V	phi	point-biserial
	ordinal	Spearman, Cramér's V	phi, Spearman, Cramér's V, gamma, tau	rank-biserial, Cramér's V	eta, tau, Spearman
	nominal	phi	rank-biserial, Cramér's V	phi, Cramér's V, lambda	rank-biserial, eta
	continuous	point-biserial	eta, tau, Spearman	rank-biserial, eta	Pearson, Spearman

3.5.9 Discretization

Some machine learning algorithm only accepts categorical features. Thus, prior transformation of numerical (continuous) variables to categorical ones is essential. Even if it can handle continuous data, a prior transformation of numerical variables

often accelerates the learning process and may produce simpler and more accurate results. Reducing the number of values for a feature is especially beneficial if a ***decision tree*** method[48] of regression or classification is to be applied to the preprocessed data because these methods are typically recursive, and a large amount of time is spent on sorting the data at each step.

As a recap, discrete and continuous data are ordinal data types with orders among the values, while nominal values do not possess any order amongst them. While the number of continuous values for an attribute can be infinitely many, the number of discrete values is often few or finite. ***Discretization*** (otherwise known as ***quantization*** or ***binning***) provides a way to partition continuous features into discrete values. Depending on how the discrete values are encoded, a numerical variable may be transformed to an ordinal one (i.e., we keep the intrinsic order of the discrete values) or to a nominal one (by using one-hot encoding, the intrinsic order between categories will be lost).

Before moving on, let's introduce some terms. ***Instance*** refers to a single collection of feature values for all features, i.e., an observation. Hereafter, N is the number of instances in the data. The term ***cut-point*** refers to a real value within the range of continuous values that divides the (sub)range into two intervals; one interval is less than or equal to the cutpoint, and the other interval is greater than the cut-point. We may also call this the ***bin edge***. The term ***arity*** in the discretization context means the number of intervals, bins, or partitions, and it is denoted here by k. Then the maximum number of cut-points is $k-1$. In general, a higher arity can make the understanding of an attribute more difficult, while a very low arity may affect predictive accuracy negatively. There are many other advantages of using discrete values over continuous ones. Discrete features are closer to a knowledge-level representation (Simon, 1981) than continuous ones. Data can also be reduced and simplified through discretization. For both users and experts, discrete features are easier to understand, use, and explain. As reported in a study (Dougherty et al., 1995), discretization makes learning more accurate and faster. [49]

Discretization helps handle outliers by placing these values into the lower or higher interval together with the remaining inlier values of the distribution. Besides, by creating appropriate bins or intervals, discretization can help spread the values of a skewed variable across a set of bins with an equal number of observations. Moreover, this transformation brings non-linearity and thus might improve the fitting power of the model. It also reduces the impact of small fluctuation in the data, which can be considered as noise. Thus, this process of "smoothing", wherein each bin smoothens fluctuations, reduces noise in the data.

Discretization shall be applied with care because some information loss occurs during this data transformation.

[48] A separate chapter is devoted to decision trees in the second volume of this book series.

3.5.9.1 Binarization

Feature ***binarization*** is the process of thresholding numerical features to get boolean values which are coded generally as 0 for `False` and 1 for `True`. Values greater than the threshold map to 1, while values less than or equal to the threshold map to 0. With a threshold of 0, only positive values map to 1. With binarization, a continuous variable will be transformed into a dichotomous one. The value of the threshold depends on the distribution of the numerical variable.

As an example, let's consider the `yr_renovated` numerical variable from the `KC-HouseSale` dataset described in Sec. 3.7.4 but already mentioned in Sec. 3.5.8.1. If the house was renovated, this variable records the year of renovation, otherwise set to zero. Because only 4.23% of the houses have been recorded as renovated, it seems natural to convert this numerical variable to a binary one by setting the threshold to 0, especially when looking at the histogram on the left of Fig. 3.47. Thus, we will have a new binary variable called `renovated`, which is set to 1 if the numerical variable contains a valid calendar year and 0 otherwise.

Table 3.34 MLR results for `KC-HouseSale` dataset with or without binarization

Input	Coefficient	Std. error	t-statistic	p-value	LCL	UCL
Intercept	-4.631e+04	4371.844	-10.594	0.000	-5.49e+04	-3.77e+04
sqft_living	278.6797	1.925	144.795	0.000	274.907	282.452
yr_renovated	80.2807	4.401	18.243	0.000	71.655	88.906
$\hat{\sigma}$=259469	R^2=0.500	(numerical)				
Intercept	-4.633e+04	4372.000	-10.597	0.000	-5.49e+04	-3.78e+04
sqft_living	278.6926	1.925	144.800	0.000	274.920	282.465
renovated	1.599e+05	8783.478	18.210	0.000	1.43e+05	1.77e+05
$\hat{\sigma}$=259476	R^2=0.500	(binary)				

In the upper part of Table 3.34, the MLR results are shown with `sqft_living` and `yr_renovated` as input numerical variables (`price` is the target). In the

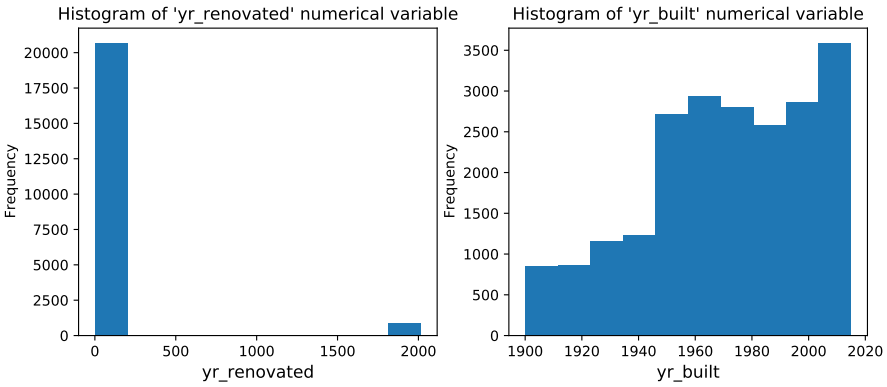


Fig. 3.47 Histogram of input numerical variables from `KC-HouseSale` dataset

3.14 Summary

In this chapter, the following items have been discussed:

- OLS usage for multiple linear regression with Advertising case study;
- Expected values and variances of OLS, respective confidence intervals and regions in case of MLR;
- Defining outliers, leverages, and influential Observations in MLR with Salary case study;
- How to assess to the goodness of fit of a regression model and what metrics can be used for measuring it;
- Graphical diagnostics as an important tool to find out outliers and influential observations;
- Linear transformations as data pre-processing (scaling and whitening transformations);
- Non-linear transformations as data pre-processing;
- Encoding non-numerical (categorical) features, respective discretizing numerical features;
- Cases when OLS assumptions are not hold, like multicoliniearity issue, multiplicative relation between predictors, polynomial regression, and weighted least squares (WLS);
- Learning theory, like the importance of bias-variance trade-off, generalization error vs. training error, respective how to estimate the generalization error in practice (training vs. testing error) with House price prediction case study.
- Feature selection using best subset selection;
- Regularization, like ridge regression, lasso regression, and elastic regression as a way for feature selection;
- Definition of hyperparameter, and ways to choose the best hyperparameter values;
- Bayesian regression;
- Multivariate linear regression.

Let's close this chapter with a good advice from John W. Tukey:

> "Hubris is the greatest danger that accompanies formal data analysis, including formalized statistical analysis. [...] Let me lay down a few basics, none of which is easy for all to accept, yet which all would be better for accepting: 1. The data may not contain the answer. The combination of some data and an aching desire for an answer does not ensure that a reasonable answer can be extracted from a given body of data." [78]

Lab Exercises

Jupyter Notebooks for the lab exercises are available at
https://github.com/FerencFarkasPhD/Advanced-Machine-Learning.

3.1. Predicting the systolic blood pressure based on the age and weight
(a) Read the observations from the `BloodPressure2` dataset introduced in Sect. 3.2 into design matrix and output vector.
(b) Plot the scatter plot of the observations together with the estimated regression plane.
(c) Calculate the hat matrix and the covariance matrices.
(d) Make hypothesis testing.
(e) Calculate the adjusted R-squared and root mean squared error.
(f) Check for collinearity.

3.2. Create a linear model to predict sales based on advertising budget
(a) Read the observations from the `Advertising` dataset introduced in Sec. 3.3.5 into design matrix and output vector.
(b) Analize the data with statistics, respective with boxplot.
(c) Check for colliniarity.
(d) Plot the scatter plot for each predictor with the estimated regression line, respective create the scatter plot matrix.
(e) Create your multiple linear regression model and draw your conclusions.
(f) Calculate the adjusted R-squared and residual squared error.
(g) Try to answer the 7 questions raised in Sec. 3.3.5 based on your results.

3.3. Identifying outliers using the leave-one-out methods
(a) Read the observations from the in the `Salary` dataset introduced in Sec. 3.4 into design matrix and output vector.
(b) After fitting the linear regression model, analyze the leverages and residuals.
(c) Check for outliers using the leave-one-out methods: DFFITS, DFBETAS, Cook's distance, and covariance ratio.
(d) Make your conclusion based on the outlier analyzes.
(e) Proceed with graphical diagnostics in outlier detection.
(f) Check the masking and swamping effect with `Star cluster` dataset.

3.4. Check how the linear transformations affect the linear regression model
(a) Read the `BloodPressure2` and `HeightWeight` datasets into design matrix and output vector.
(b) Check the distribution of the input, respective the result of the linear regression model after input is centered.
(c) Check the distribution of the input, respective the result of the linear regression model after input is standardized.
(d) Check the distribution of the input, respective the result of the linear regression model after input is min-max, respective max-abs scaled.
(e) Check how the distribution of the input and the linear regression model is changed with robust scaling.
(f) Check the effectiveness of robust scaling of the input on the `Star cluster` dataset.

3.5. Analyze the whitening transformations using different datasets
(a) Read the `Digits` dataset into an input matrix and apply a ZCA transformation.

(b) Make a visual comparison by showing the first ten digits of both the original and ZCA-whitened data.
(c) Read the `Bodyfat` dataset into design matrix and output vector.
(d) Create the scatter plot of both the original and PCA-whitened data in 3D.
(e) Create a sample from a tri-variate normal distribution with a given covariance matrix.
(f) Create the scatter plot of both the original and Cholesky-whitened data in 3D using only data within one standard deviation.
(g) Compare all 5 whitening transformations using the `Iris` dataset.

3.6. Analyze the nonlinear transformations using different datasets
(a) Using `TransistorCount` dataset, analyze the log transformation of the response variable.
(b) Using the `WordRecall` dataset, analize the log transformation of the predictor variable.
(c) Using the `Mammal` dataset, analize the log transformation of both the response and predictors.
(d) Analyze the variance stabilization transformations using the `LungCap` dataset.
(e) Analyze the power transformations with the `LungCap`, respective `red wine quality` dataset.
(f) Quantile normalizing some predictors of `Boston House-price` dataset.
(g) Normalizing data.

3.7. Encoding categorical features using different datasets
(a) Using the `BirthWeight` dataset, answer the research question: Is a baby's birth weight related to the mother's smoking during pregnancy?
(b) Measure the associations between the binary variables, respective between ordinal variables; as well as, their association with the numerical response variable, using the `KC-HouseSale` dataset.
(c) Use the nominal variable from the `Credit` dataset, and after one-hot encoding, answer the question of whether there is statistical evidence in credit balance difference between ethnicity.
(d) Use the interaction term between numerical variables, respective between numerical and categorical variables to fit a better model on the `Credit` data.
(e) Using the `SalaryDiscrimination` dataset, answer the question of whether there is statistical evidence in salary difference between genders.

3.8. Discretization of numerical variables using different datasets
(a) Binarize the `yr_renovated` numerical variable from the `KC-HouseSale` dataset.
(b) Unsupervised discretization using equal-width, equal-frequency, respective k-means binning with ordinal encoding (i.e., numerical variable dicretized into an ordinal one) using the `KC-HouseSale` dataset.
(c) Unsupervised discretization using equal-width, equal-frequency, respective k-

means binning with one-hot encoding (i.e., numerical variable dicretized into a nominal one) using the `KC-HouseSale` dataset.
(d) Supervised discretization with entropy-based binning using the `Iris` dataset.
(e) Supervised discretization with ChiMerge binning using the `Iris` dataset.

3.9. Handling missing values
(a) With deleting randomly 50% of the `CaliforniaHousing` dataset check the consequences of removing observations with missing values.
(b) With the same modified dataset described above, use simple imputation with constant, mean, median and mode and compare the RMSE score values.
(c) With the modified dataset compare the RMSE score values of the kNN imputations with different number of neighbors and with or without weighted average.
(d) With the modified dataset compare RMSE score values of the Linear regression imputation.
(e) Make a summary by comparing the goodness of fit (R-squared and MSE) of the OLS using different imputation methods.

3.10. Analyzing multicollinearity and methods to handle it
(a) Visual presentation of a multicollinearity problem.
(b) Detecting multicollinearity.
(c) Effect of (nearly) uncorrelated predictors.
(d) Effect of highly correlated predictors.
(e) Reducing data-based multicollinearity.
(f) Reducing structural-based multicollinearity.
(g) Principal Components Regression.
(h) Case study: Body fat percentage.

3.11. Fitting polynomial regression on the `Boston-House price` dataset
(a) Analyze the data graphically with box plot and scatter plot matrix.
(b) Fitting a linear model on the data and checking for correlations and outliers. Analyze the residual plot.
(c) Fit a polynomial regression using only two of the original input variables which have the highest correlation with the target. Check the results including the residual plot.
(d) Fit a polynomial regression using three of the original input variables which have the highest correlation with the target. Check the results including the residual plot.
(e) Fit a polynomial regression using four of the original input variables which have the highest correlation with the target. Check the results including the residual plot.
(f) Analyze the risk of overfitting in case of polynomial regressions.

3.12. Fitting a calibration curve on the `Line-spacing measurement` data
(a) Analyze the data graphically scatter plot matrix.
(b) Fit a linear model and check the goodness of fit scores.

(c) Check for heteroskedacticity.
(d) Try to solve the heteroskedactity problem with variance stabilizing transformations.
(e) Fit a model using weighted least squares. Weights shall be estimated with different methods described in the book.
(f) Compare the results of the OLS with WLS using different methods for weight estimation.

3.13. Analysis of autocorrelation
(a) Fitting OLS on the `Employee` dataset and check for autocorrelation in the residuals.
(b) Fit a linear model on the `Employee` using Cochrane-Orcutt and Prais-Winsten iterative estimation procedure.
(c) Autocorrelation parameter estimation with Hildreth–Lu grid search procedure.
(d) Fit a linear model that can be used to predict ice cream demand using `Icecream` dataset.

3.14. Estimation of generalization error in practice for the `KC-HouseSale` dataset
(a) Exploratory Data Analysis on the dataset.
(b) Fit a multiple linear model on the above dataset and check for multicollinearity and autocorrelation.
(c) Analyze the bias and variance of a model as model complexity changes.
(d) Analyze the training and testing error as model complexity changes.
(e) Compare the different testing methods using Monte Carlo simulations.
(f) Compare the tests results of the two model (with the original, respective with the transformed target) using different testing methods.

3.15. Feature selection with filtering and wrapper methods
(a) Feature selection based on feature importance.
(b) Filtering with low variance.
(c) Filtering with χ^2 and F-test.
(d) Filtering with mutual information between the predictors and target.
(e) Best subset selection
(f) Forward and backward stepwise selection.
(g) Stepwise selection with cross-validation.
(h) Feature selection with information criteria.

3.16. Feature shrinkage and selection with embedded methods
(a) Feature shrinkage with L2 regularization (ridge regression).
(b) Feature selection with L1 regularization (Lasso regression).
(c) Elastic net.

References

1. ***, Beware of Data Leakage, Yellowroad Blog, 2017
https://blog.myyellowroad.com/beware-of-data-leakage-f6c307009ad9
2. ***, Engineering Statistics Handbook, NIST/SEMATECH e-Handbook of Statistical Methods, U.S. Department of Commerce, 2013
https://www.itl.nist.gov/div898/handbook/index.htm
3. ***, Point-Biserial Correlation using SPSS Statistics, Laerd Statistics, Lund Research Ltd, 2018
https://statistics.laerd.com/spss-tutorials/point-biserial-correlation-using-spss-statistics.php
4. Aggarwal C. C., Outlier Analysis, IBM T. J. Watson Research Center, Yorktown Heights, New York, 2016
https://rd.springer.com/book/10.1007/978-3-319-47578-3
5. Allison, P. D., Multiple Regression: A Primer, Pine Forge Press; 1st edition, 1999.
6. Aswani, A., Engineering Statistics, Quality Control, and Forecasting, IEOR 165, Lecture notes, Berkeley University of California, 2021.
https://rd.springer.com/book/10.1007/978-3-319-47578-3
7. Bain, L. J., et al, Introduction to Probability and Mathematical Statistics, Cengage Learning, 2000
8. Beasley, T. M., et al. Rank-based inverse normal transformations are increasingly used, but are they merited?, Behavior genetics, Vol. 39(5), p. 580-595, doi:10.1007/s10519-009-9281-0, 2009
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2921808/
9. Belsley, D. A., et al, Regression Diagnostics, Identifying Influential Data and Sources of Collinearity, John Wiley & Sons, Inc., 2004
10. Bergsma, W., A bias-correction for Cramér's V and Tschuprow's T, Journal of the Korean Statistical Society, DOI: 10.1016/j.jkss.2012.10.002, 2013
https://www.researchgate.net/publication/270277061_A_bias-correction_for_Cramer's_V_and_Tschuprow's_T
11. Box, G. E. P., et al, An Analysis of Transformations, Journal of the Royal Statistical Society, Series B, Vol. 26, No. 2, 1964
https://www.researchgate.net/publication/224839719_An_Analysis_of_Transformations_with_Discussion
12. Box, G. E. P., et al, Response Surfaces, Mixtures, and Ridge Analyses, John Wiley & Sons, Inc., 2007
13. Box, G. E. P., Science and Statistics, Journal of the American Statistical Association, Vol. 71, No. 356., 1976
http://mkweb.bcgsc.ca/pointsofsignificance/img/Boxonmaths.pdf
14. Bucci, A., Cholesky-ANN models for predicting multivariate realized volatility, Università Politecnica delle Marche, MPRA Paper No. 95137, 2019
https://mpra.ub.uni-muenchen.de/95137/1/MPRA_paper_95137.pdf
15. Burrus, C. S., Iterative Reweighted Least Squares, OpenStax-CNX, 2012
https://cnx.org/exports/92b90377-2b34-49e4-b26f-7fe572db78a1@12.pdf/iterative-reweighted-least-squares-12.pdf
16. Buteikis, A., Practical Econometrics and Data Science, Vilnius University, 2020
http://web.vu.lt/mif/a.buteikis/wp-content/uploads/PE_Book/
17. Cavanaugh, J. E., et al, The Akaike information criterion: Background, derivation, properties, application, interpretation, and refinements, WIREs Computational Statistics, Wiley Interdisciplinary Reviews, DOI: 10.1002/wics.1460, 2019
18. Cook, R. D., et al, Residuals and influence in regression, New York: Chapman and Hall, 1982
https://conservancy.umn.edu/handle/11299/37076
19. Croarkin C., et al, Measurement Assurance for Dimensional Measurements on Integrated-Circuit Photomasks, U.S. Department of Commerce, National Bureau of Standards, NBS

Technical Note 1164, 1982
https://nvlpubs.nist.gov/nistpubs/Legacy/TN/nbstechnicalnote1164.pdf
20. Dougherty, J., et al, Supervised and Unsupervised Discretization of Continuous Features, ICML, pp. 194–202, 1995.
http://robotics.stanford.edu/users/sahami/papers-dir/disc.pdf
21. Dunn, K., Process Improvement Using Data, 2020.
https://learnche.org/pid/contents
22. Dunn, P., et al, Generalized Linear Models With Examples in R, Springer Texts in Statistics, 2018.
23. Elashoff, J. D., et al, Measures of Association, Stanford University, Stanford Center for Research and Development in Teaching, Research and Development Memorandum No. 93, 1972.
https://files.eric.ed.gov/fulltext/ED069788.pdf
24. Fang, S. C., et al, Entropy Optimization and Mathematical Programming, Springer Scinece+Business Media, LLC, 1997.
25. Fayyad, U. M., et al, Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning, Machine Learning, Chambery, France, 1993.
https://trs.jpl.nasa.gov/handle/2014/35171
26. Fayyad, U. M., et al, On the Handling in Decision Tree of Continuous-Valued Attributes Generation, Technical Note, Machine Learning 8, 87–102, 1992.
https://doi.org/10.1007/BF00994007
27. Gareth, J., et al., An Introduction to Statistical Learning with Applications in R, 8th Ed., Springer, 2018.
http://www-bcf.usc.edu/~gareth/ISL/
28. Glen, S., Elementary Statistics for the rest: Stepwise Regression, StatisticsHowTo.com, 2015.
https://www.statisticshowto.com/stepwise-regression/
29. Ginestet, C. E., MA 575 Linear Models, Lecture notes, Boston University, 2013.
http://math.bu.edu/people/cgineste/classes/ma575/w/index.html
30. Goodman L. A., et al, Measures of Association for Cross Classifications, Springer Series in Statistics, Springer-Verlag, 1979.
31. Göktaş, A., et al, A Comparison of the Most Commonly Used Measures of Association for Doubly Ordered Square Contingency Tables via Simulation, Metodološki zvezki, Vol. 8, No. 1, p. 17-37, 2011
https://ibmi.mf.uni-lj.si/mz/2011/no-1/goktas.pdf
32. Green H. W., Econometric Analysis, Fifth Edition, Prentice Hall, 2003.
33. Helwig, N. E., Multiple Linear Regression, Lecture slides, University of Minnesota, 2017
http://users.stat.umn.edu/~helwig/notes/mlr-Notes.pdf
34. Hunter, J. D., Matplotlib: A 2D graphics environment, Computing In Science & Engineering, vol. 9, nr. 3, pp. 90–950, 2007, COMPUTER SOC doi: 10.1109/MCSE.2007.55.
https://ieeexplore.ieee.org/document/4160265
35. Hyde, S., Likelihood based inference on the Box-Cox family of transformation: SAS and MATLAB programs, Department of Mathematical Sciences, Montana State University, 1999
https://www.jekyll.math.byuh.edu/papers/mspaper99.pdf
36. Iglewicz B., et al, Volume 16: How to Detect and Handle Outliers, The ASQC Basic References in Quality Control: Statistical Techniques, Edward F. Mykytka, Ph.D., Editor., 1993
https://hwbdocuments.env.nm.gov/LosAlamosNationalLabs/TA54/11587.pdf
37. Irizarry, R. A., Introduction to Data Science - Data Analysis and Prediction Algorithms with R, 2019.
https://rafalab.github.io/dsbook/
38. Johnson, J., et al, Econometric Methods, The McGraw-Hill Companies, Inc., Singapore, 2007.
39. Jolliffe, I. T., A Note on the Use of Principal Components in Regression, Journal of the Royal Statistical Society. Series C (Applied Statistics), Vol. 31, No. 3, pp. 300-303, 1982.
https://pdfs.semanticscholar.org/f219/2a76327e28de77b8d27db3432b6a20e7ebd7.pdf

40. Jones E., Oliphant E., Peterson P., et al., SciPy: Open Source Scientific Tools for Python, 2001.
http://www.scipy.org/
41. Kerber, R., ChiMerge: Discretization of Numeric Attributes, AAAI-92 Proceedings, 1992.
https://www.aaai.org/Papers/AAAI/1992/AAAI92-019.pdf
42. Kessy, A., et al, Optimal Whitening and Decorrelation, The American Statistician 2018, Vol. 72, No. 4, 2018.
https://arxiv.org/pdf/1512.00809.pdf
43. Korjus, K., et al, An Efficient Data Partitioning to Improve Classification Performance While Keeping Parameters Interpretable. PLoS ONE 11(8): e0161788, doi:10.1371/journal.pone.0161788, 2016.
https://www.researchgate.net/publication/307087929_An_Efficient_Data_Partitioning_to_Improve_Classification_Performance_While_Keeping_Parameters_Interpretable
44. Kraskov, A., et al, Estimating mutual information, Physical Review E 69, 066138, DOI:10.1103/PhysRevE.69.066138, 2004.
https://journals.aps.org/pre/pdf/10.1103/PhysRevE.69.066138
45. Kuhn, M., et al, Feature Engineering and Selection: A Practical Approach for Predictive Models, 2019.
https://bookdown.org/max/FES/
46. Kutner, M. H., et al, Applied Linear Statistical Models, McGraw-Hill Irwin, 2005.
47. Li, J., et al, Applied Data Mining and Statistical Learning, Department of Statistics, The Pennsylvania State University, 2018.
https://online.stat.psu.edu/stat508/
48. Li, J., et al, Applied Data Mining & Statistical Learning, STAT508 online course, Department of Statistics, The Pennsylvania State University, 2022.
https://online.stat.psu.edu/stat508/
49. Liu, H., et al, Discretization: An Enabling Technique, Data Mining and Knowledge Discovery, Nr. 6, p.393-423, 2002.
https://www.researchgate.net/publication/220451974_Discretization_An_Enabling_Technique
50. Loonis, V., et al. Handbook of Spatial Analysis, Theory and Application with R, Insee - Eurostat, Nr. 131, 2018
https://ec.europa.eu/eurostat/documents/3859598/9462709/INSEE-ESTAT-SPATIAL-ANA-18-EN.pdf
51. Max, K., et al, Feature Engineering and Selection: A Practical Approach for Predictive Models, On-line text, 2019
https://bookdown.org/max/FES/
52. Napier, G., Time Series, Lecture notes, 2020
https://bookdown.org/gary_a_napier/time_series_lecture_notes/
53. Neath, A. A., et al, The Bayesian information criterion: background, derivation, and applications, WIREs Computational Statistics, 4:199–203. doi: 10.1002/wics.199, 2012
54. Nisbet, R., et al, Handbook of Statistical Analysis and Data Mining Applications, Elsevier Inc., 2009
55. O'Brien, F. J., A Proof That t2 and F are Identical: The General Case, ERIC Number: ED215894, 1982
https://eric.ed.gov/?id=ED215894
56. Pal, K. K. at al., Preprocessing for image classification by convolutional neural networks, 2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, 2016, pp. 1778-1781, doi: 10.1109/RTEICT.2016.7808140
57. Park T., et al., The Bayesian Lasso, American Statistical Association Journal of the American Statistical Association, June 2008, Vol. 103, No. 482, Theory and Methods, DOI 10.1198/016214508000000337
https://people.eecs.berkeley.edu/~jordan/courses/260-spring09/other-readings/park-casella.pdf

58. Pedregosa F. et al., Scikit-learn: Machine Learning in Python, JMLR 12, pp. 2825-2830, 2011. http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html
59. Perez L. V., Principal Component Analysis to Address Multicollinearity, Whitman College, 2017. https://www.whitman.edu/Documents/Academics/Mathematics/2017/Perez.pdf
60. Perner P., et al., Multi-Interval Discretization Methods for Decision Tree Learning, Advances in Pattern Recognition, LNCS 1451, Springer Verlag, p. 475-482, 1998. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.88.5802&rep=rep1&type=pdf
61. Rathbun S., et al., Applied Multivariate Statistical Analysis, STAT 505 online course, Department of Statistics, The Pennsylvania State University, 2020. https://online.stat.psu.edu/stat505/
62. Robinson C., et al., Interaction Effects: Centering, Variance Inflation Factor, and Interpretation Issues, Multiple Linear Regression Viewpoints, Vol. 35(1), 2009. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.620.5853&rep=rep1&type=pdf
63. Ross B. C., Mutual Information between Discrete and Continuous Data Sets, PLoS ONE 9(2): e87357. doi:10.1371/journal.pone.0087357, 2014. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.620.5853&rep=rep1&type=pdf
64. Rousseeuw P. J., et al., Anomaly Detection by Robust Statistics, WIREs Data Mining Knowl Discov 2018, 8:e1236. doi: 10.1002/widm.1236 https://onlinelibrary.wiley.com/doi/epdf/10.1002/widm.1236
65. Rousseeuw P. J., et al., A Fast Algorithm for Minimum Covariance Determinant Estimator, Technometrics, Vol. 41, No. 3, 1998. https://wis.kuleuven.be/stat/robust/papers/1999/rousseeuwvandriessen-fastalgorithmformcd-technomet.pdf
66. Rousseeuw P. J., et al., Robust Regression and Outlier Detection, John Wiley & Sons, Inc. , 1987.
67. Shalahb, Regression Analysis, Lecture notes, Indian Institute of Technology Kanpur. http://home.iitk.ac.in/~shalab/course5.htm
68. Shalizi, C., Moving Beyond Conditional Expectations: Predictive Comparisons, Weighted Least Squares, Lecture note of Advanced Data Analysis, Carnegie Mellon University, 2013. https://www.stat.cmu.edu/~cshalizi/402/lectures/05-variance-and-weights/lecture-05.pdf
69. Shalizi, C., Tests and Confidence Sets for Multiple Coefficients, Lecture note of Modern Regression, Carnegie Mellon University, 2015. https://www.stat.cmu.edu/~cshalizi/mreg/15/lectures/18/lecture-18.pdf
70. Simon, G., Multiple Regression Diagnostics, NYU Stern, 2003. http://people.stern.nyu.edu/gsimon/Pamphlets/MultipleRegressionDiagnosticsCOLLECTION.pdf
71. Simon, L., et al., Probability Theory and Mathematical Statistics, STAT 414/415 online course, Department of Statistics, The Pennsylvania State University, 2018. https://onlinecourses.science.psu.edu/stat414/
72. Simon, L., et al., Regression Methods, STAT 501 online course, Department of Statistics, The Pennsylvania State University, 2018. https://onlinecourses.science.psu.edu/stat501/
73. Seabold, S., et al., Statsmodels: Econometric and Statistical Modeling with Python, Proceedings of the 9th Python in Science Conference (SCIPY 2010), 2010. http://conference.scipy.org/proceedings/scipy2010/pdfs/seabold.pdf
74. Storkey, A. J., When Training and Test Sets are Different: Characterising Learning Transfer, Institute of Adaptive and Neural Computation School of Informatics, University of Edinburgh, 2013. https://homepages.inf.ed.ac.uk/amos/publications/Storkey2009TrainingTestDifferent.pdf
75. Stricevic, I., et al., King County House Prices Prediction Model, 2017. https://www.slideshare.net/PawanShivhare1/predicting-king-county-house-prices
76. Tibshirani, R., et al., Modeling Basics: Assessment, Selection, and Complexity, Statistical Machine Learning Course, Carnegie Mellon University, 2015. http://www.stat.cmu.edu/~ryantibs/statml/review/modelbasics.pdf

77. Troyanskaya, O., et al., Missing value estimation methods for DNA microarrays, Bioinformatics, Vol. 17, No. 6, p. 520-525, 2001.
file:///C:/Users/bhuffark/Downloads/Missing_Value_Estimation_Methods_for_DNA_Microarra.pdf
78. Tukey, J. W., Sunset Salvo, The American Statistician, Vol. 40, No. 1, 1986.
http://www-stat.wharton.upenn.edu/~steele/HoldingPen/SunsetSalvo.pdf
79. Ventura, D., On Discretization as a Preprocessing Step For Supervised Learning Models, A Thesis Presented to the Department of Computer Science Brigham Young University, 1995.
http://axon.cs.byu.edu/papers/ventura.thesis.ps
80. Verbeek, M., A Guide to Modern Econometrics, 2nd edition, John Wiley & Sons, Ltd, 2004.
81. Wadsworth, J., Econometrics - Quantitative Methods II, Lecture notes, University of London
http://personal.rhul.ac.uk/uhte/006/ec2203/index2.html
82. Wood, F., Nonparametric Regression and Bonferroni joint confidence intervals, Lecture slides, University of Oxford, 2010
http://www.robots.ox.ac.uk/~fwood/teaching/W4315_Fall2011/Lectures/lecture_9/lecture_9.pdf
83. Zar, J. H., Spearman Rank Correlation, Lecture notes, University of Wisconsin, 2010
ftp://biostat.wisc.edu/pub/chappell/800/hw/spearman.pdf
84. Zuber, V., et al., Gene ranking and biomarker discovery under correlation, Lecture notes, Bioinformatics, Volume 25, Issue 20, Pages 2700–2707, 2009
https://academic.oup.com/bioinformatics/article/25/20/2700/193240
85. Zuber, V., et al., High-Dimensional Regression and Variable Selection Using CAR Scores, Statistical Applications in Genetics and Molecular Biology, Volume 10, Issue 1, Article 34, 2011
http://www.strimmerlab.org/publications/journals/carscore2011.pdf

Chapter 4
Time Series Analysis

4.1 Introduction to Time Series

Time series data is defined as an ordered sequence of values of a variable (series of observations) taken at successive equally spaced points in time. Needless to say, that in time series data the ordering matters. Ordering is very important because there is dependency, and changing the order could change the meaning of the data.

Time series data often arise when monitoring industrial processes, natural phenomena, or tracking corporate business metrics. Some examples of time series are: yields in a production line, heights of ocean tides, daily minimum or maximum temperature of a geographical location, and the daily closing value of the Dow Jones Industrial Average.

Time series analysis (TSA) comprises methods for analyzing time series data in order to extract meaningful statistics and other characteristics of the data. The basic objective of time series analysis is to determine a model that can describe the pattern of the time series, this is also called ***time series modeling***. Then such a model can be used:

- To describe the important features of the time series pattern.
- To explain how the past affects the future and how two time series can "interact".
- To forecast future values of the series based on past values - called ***time series forecasting***.
- To use for feedback control that can measure the quality of product in some manufacturing situations.

The natural question would arise then, why we dedicate a separate chapter for this kind of data instead of using OLS regression methods presented earlier.

> "Time series analysis accounts for the fact that data points taken over time may have an internal structure (such as autocorrelation, trend or seasonal variation) that should be accounted for." [1]

Now, let's see what exactly this internal structure would mean. With the assumption of independent error we can define both the error term and the dependent vari-

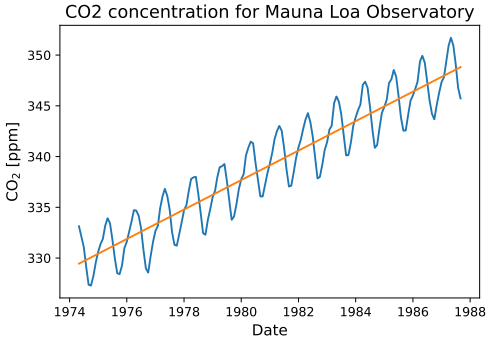


Fig. 4.1 Monthly mean CO_2 concentrations measured at the Mauna Loa Observatory from 1974 to 1987

able as a random variable with some mean and standard deviation. In time series data we can also define statistical moments like mean and variance, but an important feature of many time series process is that their mean and/or variance change over time.

On Fig. 4.1 we can see the monthly mean CO_2 concentrations[1] measured at the Mauna Loa Observatory from 1974 to 1987, which is a good example of a time series data with constant variance, but varying mean. We say that there is an increasing ***trend*** in the CO_2 mean concentration over time. This time series data has another interesting feature, namely the CO_2 concentration varies periodically, maximum values are around May and minimum values around November. We say that is has a seasonal component.

In Fig. 4.2 the daily return[2] of the Nasdaq composite index[3] is shown which has constant mean of zero, but with changing variance ("volatility" of the daily return changes over time).

Similarly to linear regression, time series modeling allow us to replicate each observation by decomposing the underlying process into a linear combination of explanatory variables plus the random noise.

Before we apply a model to a time series we need to pre-process the data in order to decompose into its components part:

- de-trending (non-stationarity, seasonality)
- autocorrelation
- outliers
- "low-pass filtering"[4] (moving average, exponential smoothing)

[1] The dataset can be downloaded from https://www.itl.nist.gov/div898/handbook/datasets/MLCO2MON.DAT.

[2] Daily return is the gain or loss of a portfolio and is calculated by subtracting the opening price from the closing price, and the result is divided by the opening price. Finally, the result is multiplied by 100 to convert to percentage.

[3] You can download the data set from https://finance.yahoo.com/quote/%5EIXIC/history/.

[4] In electrical engineering term a low-pass filter is a filter which allows low-frequency (long-period) variations to "pass-through" the filter, while the high-frequency (short-period) variations are reduced.

A ***univariate time series*** is a sequence of measurements of the same variable collected over time. Most often, the measurements are made at regular time intervals.

If the records contains time series of more than one variable then we have a ***multivariate time series***.

A time series model is said to be linear or non-linear depending on whether the current value of the series is a linear or non-linear function of past observations. In a continuous time series observations are measured at every instance of time, whereas a discrete time series contains observations measured at discrete points in time. In practice we usually have ***discrete time series***.

In general, a time series is affected by four components:

- ***trend***, denoted with T(t) - The general tendency of a time series to increase, decrease or stagnate over a long period of time.
- ***seasonal***, denoted with S(t) - This component explains fluctuations within a year during the season, usually caused by climate and weather conditions, customs, traditional habits, etc.
- ***cyclical***, denoted with C(t) - This component describes the medium-term changes caused by circumstances, which repeat in cycles, and it is different from seasonal variation. The duration of a cycle extends over longer period of time, usually expanding over many years. In practice, the trend component is assumed to include also the cyclical component.
- ***residuls***, denoted with R(t) - component with irregular variation that is left after other components have been calculated and removed from time series data Random variations are caused by unpredictable influences, for example due to incidences like revolutions, strikes, or natural disasters, etc.

Considering the effects of these four components, two different types of models are generally used for a time series.

- Additive model: $X_t = T(t) + S(t) + C(t) + R(t)$ - time series X_t is formed by the sum of time series components and it is assumed that these four components are independent of each other. An additive model is appropriate if the magnitude of the seasonal fluctuations does not vary with the level of time series;

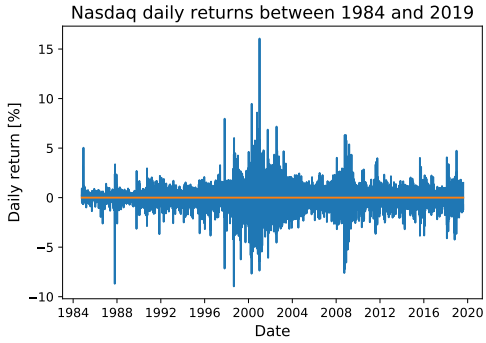


Fig. 4.2 daily return of the Nasdaq composite index

- multiplicative model: $X_t = T(t) \cdot S(t) \cdot C(t) \cdot R(t)$ - time series X_t is formed by the product of time series components and it is assumed that these four components are not necessarily independent and they can affect one another. The multiplicative model is appropriate if the seasonal fluctuations increase or decrease proportionally with increases and decreases in the level of the series

In both additive and multiplicative cases the time series X_t is called a trend stationary (TS) series, that is, after removing the deterministic part from a TS series, what remains is a stationary series.

Multiplicative decomposition is more prevalent with economic series because most seasonal economic series do have seasonal variations which increase with the level of the series. Rather than choosing either an additive or multiplicative decomposition, we could transform the data beforehand.

Very often the transformed series can be modeled additively when the original data is not additive. In particular, logarithms turn a multiplicative relationship into an additive relationship. If our model is $X_t = T(t) \cdot S(t) \cdot C(t) \cdot R(t)$ then taking the logarithms of both sides gives us:

$$\log X_t = \log T(t) + \log S(t) + \log C(t) + \log R(t) \tag{4.1}$$

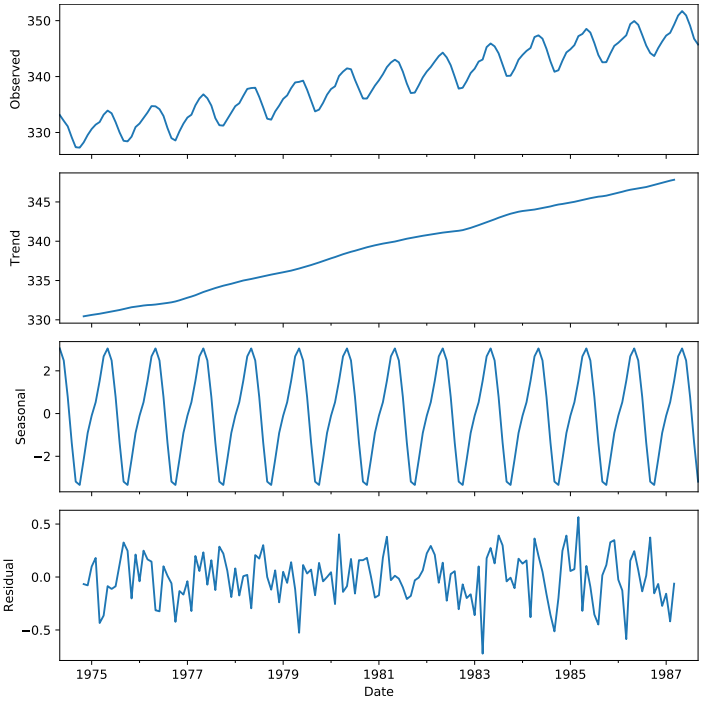


Fig. 4.3 Decomposition of CO2 time series

So, we can fit a multiplicative relationship by fitting a more convenient additive relationship to the logarithms of the data and then to move back to the original series by exponentiating.

Estimate the trend. Two approaches:

1. Using a smoothing procedure
2. Specifying a regression equation for the trend

De-trending the series: 1. For an additive decomposition, this is done by subtracting the trend estimates from the series; 2. For a multiplicative decomposition, this is done by dividing the series by the estimated trend values.

Estimating the seasonal factors from the de-trended series: Calculate the mean (or median) values of the de-trended series for each specific period (for example, for monthly data - to estimate the seasonal effect of January - average the de-trended values for all January values in the series etc). The number of seasonal factors is equal to the frequency of the series (e.g. monthly data = 12 seasonal factors, quarterly data = 4, etc.).

The seasonal effects should be normalized: 1. For an additive model, seasonal effects are adjusted so that the average of d seasonal components is 0 (this is equivalent to their sum being equal to 0); 2. For a multiplicative model, the d seasonal effects are adjusted so that they average to 1 (this is equivalent to their sum being equal to d);

Analyze the residual component. Whichever method was used to decompose the series, the aim is to produce stationary residuals. Choose a model to fit the stationary residuals (e.g. see ARMA models). Forecasting can be achieved by forecasting the residuals and combining with the forecasts of the trend and seasonal components.

One of the fundamental difference between time series and random sampling from a single, knowing distribution is that the time series often have a "memory" - that is, value of a given time period is influenced by previous time periods. This memory is quantified with the notion of autocorrelation.

The aim of the "low-pass filtering" is to remove small (low amplitude) events which occur more frequently, and thus only big (high amplitude) events, that do not occur very often, will remain in the time series data.

The usage of time series modeling is twofold:

- Obtain an understanding of the forces and structure that produced the observed time series data
- fit a model in order to be able to forecast, monitor or even provide feedback control.

==========

In linear regression we assumed that the independent variable is not random, only the dependent variable. In time series data the independent variable is the time, while the observations are the realization of a random variable. The problem is, that one of the OLS assumption restricts the presence of auto-correlation between the errors. In time series data, however, most often there is some correlation between adjacent

observations. Successive observations are usually correlated, so future values may be predicted from past values, which is called ***forecasting***.

Note *Time series shall be seen as a realization of a stochastic process, i.e. see the time series dataset as the actual values of a random variable over time. This random variable can be characterized with a distribution and moments, like mean and variance.*

Now, let's assume that you have an ordered time series data taken at equal time interval, a total of N observations. In order to emphasize that this is not an ordinary, but a time series data we will denote the time series with X_t (that is, a random variable X dependent on time t). Then the N observations of a time series data are the realizations of this random variable X_t:

$$X_t : \quad x^{(1)}, x^{(2)}, \ldots, x^{(N)} \tag{4.2}$$

The period between two sequential observations $x^{(i)}$ and $x^{(i+1)}$ is a unit of time: hours, days, weeks, months, quarters, years, etc. - or one may be even collecting observations in other domain as well, in another kind of "distance".

Previous values of a time series are called ***lags***. The first lag of X_t is X_{t-1}, while the k-th lag is X_{t-k}. We can easily create the first lags of the N observations by shifting backward the values with one time interval.

$$X_{t-1} : \quad \big(x^{(0)},\big) x^{(1)}, \ldots, x^{(N-1)} \tag{4.3}$$

where the first element $x^{(0)}$ is placed in brackets because that is not available from the sample with N observations.

Now, the shifted data can be seen as the observations of another random variable, thus we can define the covariance between the random variable X_t and its lag k, denoted with X_{t-k}, in the same way as we did for two random variables X and Y in Eq. 2.8. The covariance between X_t and its k-th lag, X_{t-k}, is called the k-th ***autocovariance*** of the series X_t:

$$\gamma_k = \mathbb{C}[X_t, X_{t-k}] \tag{4.4}$$

By definition we have:

$$\gamma_0 \equiv \mathbb{C}[X_t, X_t] = \mathbb{V}[X_t] \tag{4.5}$$

The k-th ***autocorrelation coefficient***, also called the ***serial correlation coefficient***, measures the correlation between x_t and x_{t-k}:

$$\rho_k = \frac{\mathbb{C}[X_t, X_{t-k}]}{\sqrt{\mathbb{V}[X_t]}\sqrt{\mathbb{V}[X_{t-k}]}} \tag{4.6}$$

and with the assumption that the variance is constant over time (i.e. $\mathbb{V}[X_{t-k}] = \mathbb{V}[X_t]$=$\gamma_0$) we also have $\rho_k = \frac{\gamma_k}{\gamma_0}$.

Chapter 5
Optimization Methods

5.1 Introduction to Optimization

Because there are several optimization algorithms, we should have a deeper understanding of how they are constructed. Moreover, one should know which algorithms to use for a given application and dataset size. Thus, in this chapter, different optimization methods are discussed, focusing on those used in linear and logistic regressions.

5.1.1 Reason for Using Optimization in Machine Learning

As a reminder, the analytical solution of the multiple linear regression is provided by Eq. 3.12, and the ***computational complexity*** of the OLS with an $N \times D$ design matrix can be calculated by adding up the following numbers[1]:

- matrix multiplication $\dot{\mathbf{X}}^\top \dot{\mathbf{X}}$ takes $\mathcal{O}(N \cdot D^2)$ times
- matrix inversion $(\dot{\mathbf{X}}^\top \dot{\mathbf{X}})^{-1}$ takes $\mathcal{O}(D^3)$ times
- matrix multiplication $\dot{\mathbf{X}}^\top \mathbf{y}$ takes $\mathcal{O}(N \cdot D)$ times
- final matrix multiplication $(\dot{\mathbf{X}}^\top \dot{\mathbf{X}})^{-1} \dot{\mathbf{X}}^\top \mathbf{y}$ takes $\mathcal{O}(D^2)$ times

where the individual elements of complexity $\mathcal{O}(1)$ represents a floating-point arithmetic. Thus, the total computational complexity of the OLS is $\mathcal{O}(N \cdot D^2 + D^3 + N \cdot D + D^2)$. Since, $N \cdot D < N \cdot D^2$ and $D^2 < D^3$ for $N, D > 1$, when N and D becomes large, asymptotically the OLS computational complexity will become $\mathcal{O}(N \cdot D^2 + D^3)$. Moreover, if $N \gg D$ we approximately have $\mathcal{O}(N \cdot D^2)$.

Thus, when the dataset is large, and the number of input variables is also significant, the computational cost, both in time and memory storage, of calculating the estimated regression coefficients is outstandingly enormous. In such a case, we might be interested in finding an alternative way to obtain the estimated regression coefficients with less computational complexity. Moreover, as we have seen with

[1] See wikipedia for matrix operation complexity at https://en.wikipedia.org/wiki/Computational_complexity_of_mathematical_operations#Matrix_algebra.

Lasso regression discussed in Sec. 3.9.2, we might not even have a closed-form solution to our problem. In such a case, we are also forced to find alternative ways to obtain the optimal solution to our initial problem of minimizing the residual sum of squares with respect to a constraint. Finally, an optimization algorithm might be used even with datasets where $N < D$.

5.1.2 Optimization Algorithms

As we had pointed out several times in the book, a machine learning algorithm is nothing else than the translation of a real problem into a crunch of numbers, called a dataset, that is used to fit a function serving both for a better understanding of the real problem, as well as, to make predictions for future data. We have also seen, that in the case of multiple regression, the data is fit well (i.e., $y^{(i)} = f(\mathbf{x}^{(i)}) + \varepsilon^{(i)}$ with sufficiently small $\varepsilon^{(i)}$ for every $i = \overline{1,N}$) when the residual sum of squares are minimized. Thus, with an optimization problem, we want to find out the minimum of the cost function (or objective function) $J(\mathbf{w})$ introduced in Sec. 3.3.7.

Optimization algorithms are iterative in nature and begin with an initial guess of the variable $\mathbf{w}$ and generate a sequence of improved estimates (called "iterates") until they terminate, hopefully at a solution (we say that the optimization algorithm converged to a solution). The strategy used to move from one iteration to the next distinguishes one algorithm from another. Most strategies make use of the values of the objective function $J(\mathbf{w})$ and possibly the first and - in some cases - the second derivatives of these functions. Some algorithms accumulate information gathered at previous iterations, while others use only local information obtained at the current iteration. [6]

We may distinguish between first-order (when the first derivative is used), respective second-order (when both first and second-order derivatives are used) optimization algorithms. Most of them use the line search approach, which first finds a descent direction (the "line") along which the objective function will be reduced and then computes a step size that determines how far it should move along that direction (i.e., "searching" along that "line"). The descent direction can be computed by various methods discussed in this chapter. The step size can be determined either exactly or inexactly.

A good optimization algorithm should possess the following properties: [6]

- Robustness: it should perform well (i.e., converges to a solution) on a wide variety of problems in its class (for all reasonable values of the initial guess).
- Efficiency: it should not require excessive computer time or storage.
- Accuracy: it should be able to identify a solution with precision (without being overly sensitive to errors in the data or the arithmetic rounding errors).

different optimization algorithms will perform better or worse in different situations. Thus, studying your initial problem through an optimization lens can actually give you a deeper understanding of the task at hand.

5.2 Gradient Descent

Gradient descent (GD) is an iterative first-order optimization algorithm used to find the minimum of the objective function. Due to its importance and ease of implementation, this algorithm is commonly used in machine learning and deep learning. Gradient descent algorithm requires that the objective function is differentiable and convex[2]. So what is the gradient?

> "A gradient measures how much the output of a function changes if you change the inputs a little bit." - Lex Fridman (MIT)

Intuitively, in the case of a univariate function $J(w)$, the gradient is simply the first derivative of $J(w)$ at a selected point w_o, which represents the slope of the function $J(w)$ at that point. Now, with a random initial value of the parameter w, at each iteration, you change the value of the w such that if you make your step sizes (changes in w) proportional to the slope then, when the slope is flattening out towards the minimum of $J(w)$, your steps get smaller and smaller until w converges to its optimal value. It is also possible to use quasi-convex functions with a gradient descent algorithm. However, often they have so-called saddle points where the algorithm can get stuck (as the derivative is zero) or find a local minimum instead of the global minimum, as shown in Fig. **??**.

Now, in the case of multiple regression, the objective function is multivariate (there is more than one parameter w), so the gradient is a vector of partial derivatives in each main direction (along the coordinate axes). For example, with an objective function with two parameters $\mathbf{w} = \begin{bmatrix} w_1 & w_2 \end{bmatrix}^\top$, the graph of $J(\mathbf{w})$ represents a surface. Now, instead of asking about the slope of the function, you have to ask about which direction you should step in this input space of the parameters w_1, w_2 to decrease the value of the objective function most quickly. In other words, what is the downhill direction in which a ball would roll down the fastest? The gradient of a function gives you the direction of the steepest ascent (i.e., which direction should you step to increase the function most quickly). Then taking the negative of that gradient gives you the direction to step that decreases the function most quickly, and the length of this gradient vector is an indication of just how steep that steepest slope is[3].

There are three variants of gradient descent, which differ in how much data we use to compute the gradient of the objective function. Depending on the amount of data, we make a trade-off between the accuracy of the parameter update and the time it takes to perform an update.

---

[2] See wikipedia at https://en.wikipedia.org/wiki/Convex_function.

[3] If you are not familiar with multivariate calculus, you may want to check the first part of the video from 3Blue1Brown at https://youtu.be/IHZwWFHWa-w. Although that video starts with a brief explanation of the neural network, linear and logistic regression can be seen as the input layer of a neural network (of course, linear regression would have an identity function instead of the sigmoid). The handwritten digit recognition dataset mentioned in the video is also discussed in this book at Sec. 6.10.3.

Vanilla gradient descent, aka batch gradient descent, computes the gradient of the cost function w.r.t. to the parameters theta for the entire training dataset:

Stochastic gradient descent (SGD) in contrast performs a parameter update for each training example.

Mini-batch gradient descent finally takes the best of both worlds and performs an update for every mini-batch of n training examples.

5.2.1 Basics of Gradient Descent

For simplicity, let's consider the quadratic cost function $J(w) = w^2$ where there is only one parameter w to be determined. Thus, we need an efficient searching algorithm for finding the optimal value of w for which the cost function $J(w)$ will be minimized. We know from calculus that the first derivative of a function $J(w)$ at a given point w_o provides the rate of change of the function at that point.

$$\begin{cases} \dfrac{\mathrm{d}J(w_0)}{\mathrm{d}w} > 0 & \Rightarrow \quad J(w_o - \xi) < J(w_o) < J(w_o + \xi) \\ \dfrac{\mathrm{d}J(w_0)}{\mathrm{d}w} < 0 & \Rightarrow \quad J(w_o - \xi) > J(w_o) > J(w_o + \xi) \end{cases} \tag{5.1}$$

for arbitrary small strictly positive ξ. We aim to decrease the value of $J(w)$, so we need to add ξ for negative and subtract ξ for a positive derivative). Thus, the sign of the derivative can be used for updating the value of w. Moreover, the absolute value of the derivative also provides information about how fast we "descend" in that direction, so the magnitude of the derivative can be also used in the algorithm directly. Then in each iteration, we may update the weight w:

$$w^{(k+1)} = w^{(k)} - \alpha \frac{\mathrm{d}J(w^{(k)})}{\mathrm{d}w} \tag{5.2}$$

with $w^{(k)}$ representing the weight parameter value in the k-th iteration, and $w^{(k+1)}$ the next parameter value. There is a new parameter α, called the ***learning rate***, which controls at each iteration the size of the step we move away from the actual value of w.

Choosing the right value for the α learning rate is not straightforward and requires a lot of experimenting. If the learning rate is too low, the convergence will be very slow (more iterations are needed until an optimal solution is found). If the learning rate is too high, divergence could happen (instead of decreasing, the cost function will increase to infinity). To have a better understanding of the effect of the learning rate on the gradient descend algorithm, let's check our simple example of quadratic cost function $J(w) = w^2$ for which the updating in each iteration is $w^{(k+1)} = w^{(k)} - \alpha 2w$ and the algorithm stops when either the change in the cost function between iterations is less than 0.01 or the number of iterations reaches 20.

As can be seen in Fig. 5.1, for a small learning rate ($\alpha = 0.1$) the gradient descend algorithm converges much slower, the number of iteration is 18, compared to the case when the learning rate is slightly higher ($\alpha = 0.33$) in which case the num-

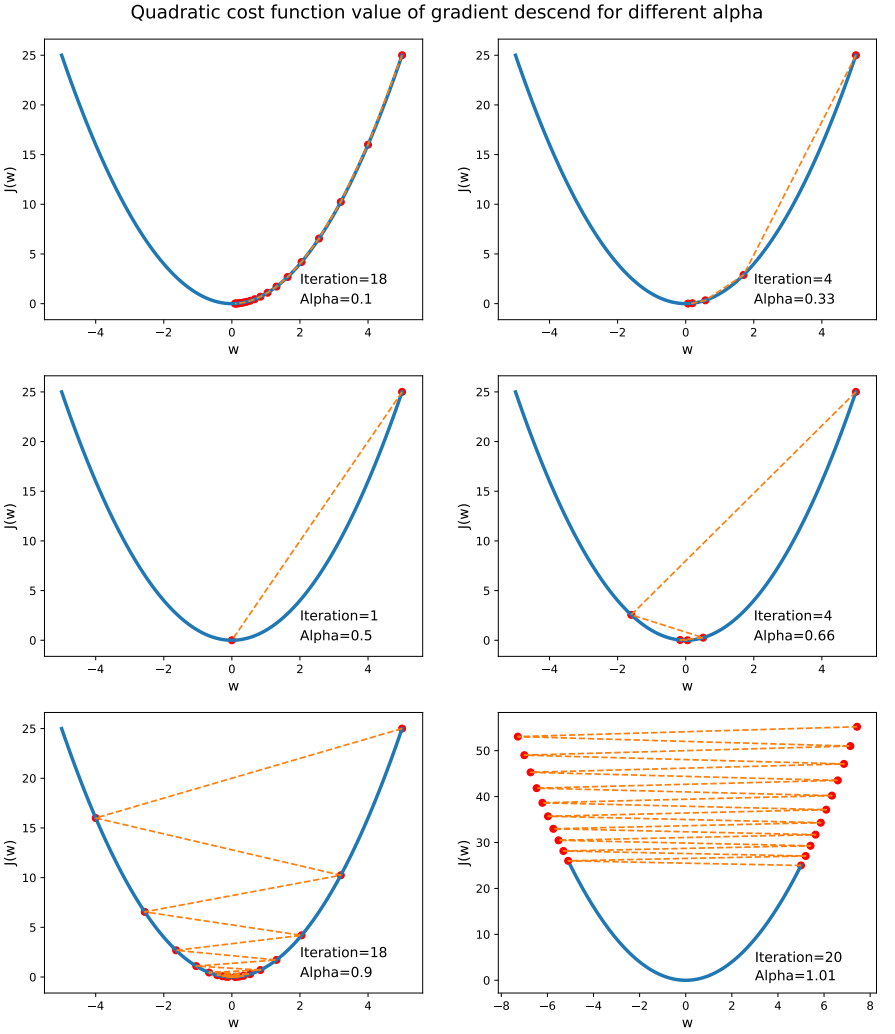


Fig. 5.1 The output as probabilities of the softmax, the hardmax and the normalized min-max scaling function for 20 linearly spaced values in the range $[-20, +20]$

ber of iterations is only 4. If the learning rate increased further and becomes greater than 0.5 the gradient descend algorithm starts oscillating and the higher the learning rate the greater the oscillation becomes. If $\alpha > 1$ the cost function will not converge to its global minimum ($J(0) = 0$), but rather diverges and goes to infinity (overflow or NaN error during program execution). Interesting to note, that as the cost function approaches its minimum, gradient descent will automatically take smaller steps without changing the value of the learning rate. In practical examples, the value of the learning rate would be much smaller than what we ahve seen in this very simple example.

In conclusion, the value of the learning rate α needs to be tuned, so this is a new parameter which optimal value shall be found out during the learning process. However, because this is not part of our model parameter w, it represents a hyper-parameter. The name of the gradient descend is self-explanatory, as the algorithm will move on the (hyper)surface of the cost function in the direction of the gradient providing the deepest descend.

5.2.2 Batch Gradient Descend

In case of multiple linear regression the number of features is D and the number of model parameters is $D+1$. In this case the cost function has a hypersurface in the $D+1$ dimensions and the gradient of the cost function $J(\mathbf{w})$ is a $D+1$ dimensional vector of partial derivatives:

$$\nabla_{J(\mathbf{w})} = \begin{bmatrix} \frac{\partial J(\mathbf{w})}{\partial w_0} \\ \frac{\partial J(\mathbf{w})}{\partial w_1} \\ \vdots \\ \frac{\partial J(\mathbf{w})}{\partial w_D} \end{bmatrix} \tag{5.3}$$

and the direction of steepest descend is given by the summation of directions provided by the partial derivatives. The gradient descend algorithm repeatedly takes a step in the direction of steepest decrease of the cost function with updating the model parameters $\mathbf{w}$:

$$\mathbf{w}^{(k+1)} = \begin{bmatrix} w_0^{(k+1)} \\ w_1^{(k+1)} \\ \vdots \\ w_D^{(k+1)} \end{bmatrix} = \begin{bmatrix} w_0^{(k+1)} - \alpha \frac{\partial J(\mathbf{w}^{(k)})}{\partial w_0} \\ w_1^{(k+1)} - \alpha \frac{\partial J(\mathbf{w}^{(k)})}{\partial w_1} \\ \vdots \\ w_D^{(k+1)} - \alpha \frac{\partial J(\mathbf{w}^{(k)})}{\partial w_D} \end{bmatrix} = \mathbf{w}^{(k)} - \alpha \cdot \nabla_{J(\mathbf{w}^{(k)})} \tag{5.4}$$

where $\mathbf{w}^{(k)}$ represents the value of the model parameters (called weights) at k-th iteration, while $\mathbf{w}^{(k+1)}$ the new parameter values.

In case of ***batch gradient descend*** the update is provided only after processing all N observations from the sample (or in other words, all samples from the training set). With a cost function of residual sum of squares (i.e. $J(\mathbf{w}) = \text{RSS}$) used in multiple linear regressions the update algorithm can be written in vectorized form using the left side of Eq. 3.11 which represents the gradient of the cost function without multiplied by (-2) (the number 2 can be built into the learning rate α):

$$\mathbf{w}' = \mathbf{w} - \alpha \cdot \mathbf{X}^\top (\mathbf{X} \cdot \mathbf{w} - \mathbf{y}) \tag{5.5}$$

An iteration through which all samples from the training set is used to update the model parameters is called an ***epoch***.

The IRLS is also used in logistic regression.

TO BE CONTINUED ...

5.7 Summary

In this chapter, the following items have been discussed:

- Fundamentals of convex optimizations
- Batch or Full Gradient Descend, respective the Stochastic Gradient Descent algorithm
- Variance reduction techniques like SAG and SAGA for the SGD
- Coordinate Descend as simpler, yet efficient algorithm used for LASSO
- Newton's methods and the Newton conjugate gradient (Newton-CG) algorithm
- Quasi-Newton methods like LBFGS, which estimate the second order derivative from the changes of the gradient
- Interactively Re-weighted Least Squares (IRLS) used in WLS, respective in logistic regression

Lab Exercises

Jupyter Notebooks for the lab exercises are available at
https://github.com/FerencFarkasPhD/Advanced-Machine-Learning.

5.1. Gradient descent optimization algorithm
(a) xxx.

References

1. Boyd, S., et al, Convex Optimization, Cambridge University Press, 2004.
https://web.stanford.edu/~boyd/cvxbook/
2. Defazio, A., et al, SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives, Advances in Neural Information Processing Systems (NIPS), 2014.
https://www.di.ens.fr/~fbach/Defazio_NIPS2014.pdf
3. Gower, R. M., Optimization for Machine Learning, Lecture notes, Cornell University, 2020.
https://gowerrobert.github.io/#teaching
4. Hunter, J. D., Matplotlib: A 2D graphics environment, Computing In Science & Engineering, vol. 9, nr. 3, pp. 90–950, 2007, COMPUTER SOC doi: 10.1109/MCSE.2007.55.
https://ieeexplore.ieee.org/document/4160265
5. Jones E., et al., SciPy: Open Source Scientific Tools for Python, 2001.
http://www.scipy.org/
6. Nocedal, J., et al, Numerical Optimization, 2nd edition, Springer Science+Business Media, 2006.
7. Ravikumar, P., Convex Optimization, Lecture notes, Carnegie Mellon University, 2017.
https://www.cs.cmu.edu/~pradeepr/convexopt/

8. Schmidt, M., et al., Minimizing Finite Sums with the Stochastic Average Gradient, Cornell University, arXiv:1309.2388 [math.OC], 2015.
https://arxiv.org/abs/1309.2388
9. Schmidt, M., Machine Learning, CPSC 540, Lecture notes, The University of British Columbia, 2018.
https://www.cs.ubc.ca/~schmidtm/Courses/540-W18/
10. Pedregosa, F., et al., Scikit-learn: Machine Learning in Python, JMLR 12, pp. 2825-2830, 2011.
http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html
11. Tibshirani R., Convex Optimization, Machine Learning 10-725, Lecture notes, Carnegie Mellon University, 2019.
https://www.stat.cmu.edu/~ryantibs/convexopt/
12. Vishnoi, N. K., Algorithms for Convex Optimization, Cambridge University Press, 2020.
https://convex-optimization.github.io/ACO-v1.pdf
13. Wang, Y., Convex optimization: Gradient Methods and Online Learning, CS292A Lecture notes, Computer Science Department, University Of California, Santa Barbara, 2019.
https://sites.cs.ucsb.edu/~yuxiangw/classes/CS292A-2019Spring/
14. Yann, N. D., et al., Identifying and attacking the saddle point problem in high-dimensional non-convex optimization, Cornell University, arXiv:1406.2572 [cs.LG], 2014.
https://arxiv.org/pdf/1406.2572.pdf

Chapter 6
Logistic Regression

6.1 Introduction

In linear regression the output is continuous and the aim is to fit a hyperplane on the input data. As we have seen, however, there are cases where the predictors are rather qualitative instead of quantitative. We discussed this extensively in Sec. 3.5.8. Now, if the predictor can be qualitative why we should limit ourselves to only quantitative response. There are many situations where we are interested in input-output relationships, as in case of linear regression, but the output variable is qualitative rather than quantitative. This is called ***classification***, and is an important topic in statistics and machine learning.

In particular there are many situations where we have binary outcomes, like `Yes`/`No`, `Healthy`/`Sick`, or `Pass`/`Fail`. So we are aiming to build a model which guesses the binary output from the input variables. But, as in case of linear regression model we assume that there is no perfect rule, and there is some noise we need to take into account, in which case there is no perfect either "yes" or "no" answer. In such case we are aiming to establish rather the probabilities of these binary outcomes. More formally, we are looking for the conditional probabilities for every value of the response variable, or more precisely the probability distribution of the response Y, given the input variables, i.e., $\mathbb{P}(Y|X)$. This would tell us about how precise our predictions are. These kind of classification problems, where the output has only binary values, are called ***binary classification***.

The ***logistic regression***, or ***logit regression***, or logit model is a regression model where the output is categorical. Thus, logistic regression is used for classification.

There are many possible classification techniques, or classifiers, that one classifier might use to predict a qualitative response.

In this chapter, we study approaches for predicting qualitative responses, a process that is known as classification. Predicting a qualitative response for an observation can be referred to as classifying that observation, since it involves assigning the observation to a category, or class. On the other hand, often the methods used for classification first predict the probability of each of the categories of a qualitative

variable, as the basis for making the classification. In this sense they also behave like regression methods.

Just as in the regression setting, in the classification setting we have a set of training observations $\mathcal{D}_{N,D} = ((\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots, (\mathbf{x}^{(N)}, y^{(N)}))$ that we can use to build a classifier. We want our classifier to perform well not only on the training data, but also on test observations that were not used to train the classifier.

6.2 Why not approach classification through regression?

The natural question would arise, why not apply the linear regression when it comes to classification, i. e. when the output is not numerical, but a categorical variable? For this reason, let's take the so-called `Iris` flower dataset introduced by the British statistician and biologist Ronald Fisher in his 1936 paper "The use of multiple measurements in taxonomic problems" [5]. The dataset consists of 50 samples from each of three species of Iris flower (Iris setosa, Iris virginica, and Iris versicolor). Thus, the output, called target, has three categories: `setosa`, `virginica`, and `versicolor`. Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters - see Table 6.1.

In Fig. 6.1, the four features are shown on an iris Versicolor flower. Our task is to classify a given iris flower intro three species based on these four features: `sepal lenght`, `sepal width`, `petal lenght`, and `petal width`. In this simple dataset with $N = 150$ observations, we want to build a model which could classify the iris flower into three species based on these four features. In Sec. 3.5.8, we have already seen how categorical variables can be encoded. We may try to encode the class labels as a quantitative response variable y as already shown in the `target` column of Table 6.1. Then using this coding, OLS could be used to fit a linear regression model to predict y based on the four predictors listed in Table 6.1. In

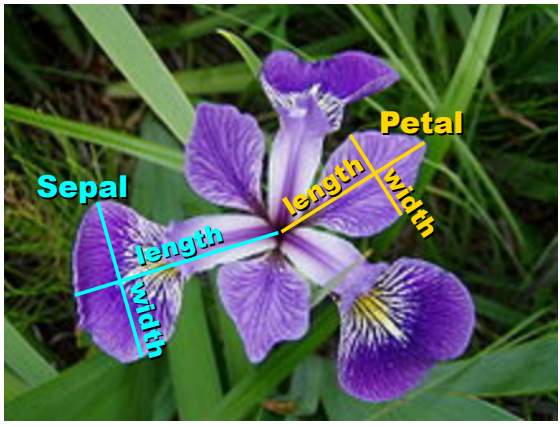


Fig. 6.1 Identifying the features of the `Iris` dataset on an iris versicolor (image of the iris flower from Wikipedia)

order to visualize this, let's use the PCR already described in Sec. 3.6.1.4 using only the first principal component. The result can be seen in Fig. 6.2.

There are two problems, though. Unfortunately, this coding implies an ordering on the outcomes, putting `versicolor` in between `setosa` and `virginica` iris flower, and insisting that the difference between `setosa` and `versicolor` is the same as the difference between `versicolor` and `virginica`. In practice, however, we may choose a different coding like, giving 2 for `setosa`, 0 for `versicolor`, and 1 for `virginica`, which would imply a totally different relationship among the three species. Each of these codings would produce fundamentally different linear models that would ultimately lead to different sets of predictions on future observations.

Table 6.1 A sample of `Iris` dataset (first three observations from each species)

Nr.	sepal length x_1	sepal width x_2	petal length x_3	petal width x_4	target y	class label
1	5.1	3.5	1.4	0.2	0	setosa
2	4.9	3.0	1.4	0.2	0	setosa
3	4.7	3.2	1.3	0.2	0	setosa
...	...	...	...	...	...	...
51	7.0	3.2	4.7	1.4	1	versicolor
52	6.4	3.2	4.5	1.5	1	versicolor
53	6.9	3.1	4.9	1.5	1	versicolor
...	...	...	...	...	...	...
101	6.3	3.3	6.0	2.5	2	virginica
102	5.8	2.7	5.1	1.9	2	virginica
103	7.1	3.0	5.9	2.1	2	virginica
...	...	...	...	...	...	...

The other problem is that we are aiming for probabilities, which this linear model does not provide. One might observe, that if we take only two classes, like `setosa` coded as 0 and `versicolor` coded as 1, then we could fit a linear regression to this binary response, and predict `versicolor` when $y \geq 0.5$ and `setosa` otherwise.

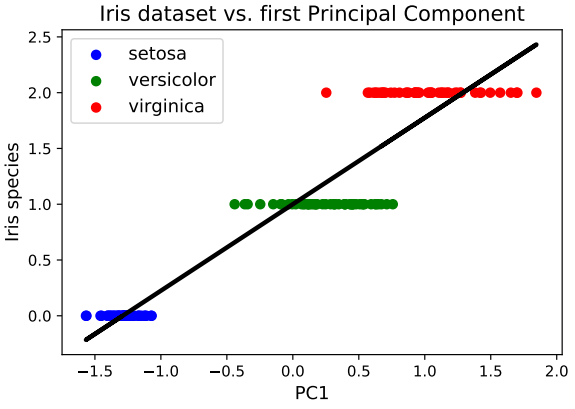


Fig. 6.2 Applying Principal Components Regression (PCR) on the `Iris` dataset using the first principal component only

In the binary case, it is not hard to show that, even if we flip the above coding, linear regression will produce the same final predictions. For a binary response with a 0/1 coding as above, regression by least squares does make sense; it can be shown that the $\dot{\mathbf{X}} \cdot \mathbf{w}$ obtained using linear regression is an estimate of $\mathbb{P}(\texttt{versicolor}|\mathbf{X})$ in this special case [6]. However, if we use linear regression, some of our estimates might be outside the [0, 1] interval, as can be seen in Fig. 6.3, making them hard to interpret as probabilities!

6.3 Binomial Logistic Regression

In linear regression the aim was to fit a hyperplane on the dataset, that is, a linear model of the form $\hat{y} = h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$, and try to predict the value of a new y for a given x. We already see examples when one of the features was a categorical variable. Now, let's see the case when not only the input but the output can be also be a categorical variable. For the moment we discuss only the binary case, when the output can have only two values (without loss of generality we say that y can be either 0 or 1).

6.3.1 Problem Formulation

In logistic regression we try to predict the probability that a given observation belongs to the first class versus the probability that it belongs to the second class. Specifically, we will try to learn a function of the form:

$$\begin{cases} \mathbb{P}(y^{(i)} = 1|\mathbf{x}^{(i)}) = \dfrac{1}{1+\exp(-\mathbf{w}^\top \mathbf{x}^{(i)})} = \sigma(\mathbf{w}^\top \mathbf{x}^{(i)}) \\ \mathbb{P}(y^{(i)} = 0|\mathbf{x}^{(i)}) = 1 - \mathbb{P}(y^{(i)} = 1|\mathbf{x}^{(i)}) = 1 - \sigma(\mathbf{w}^\top \mathbf{x}^{(i)}) \end{cases} \tag{6.1}$$

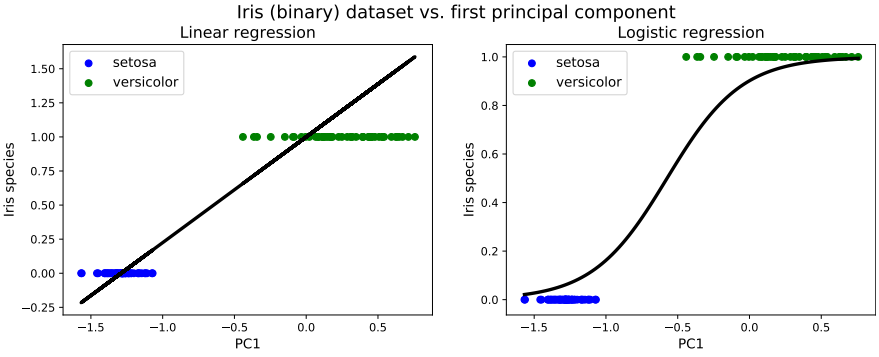


Fig. 6.3 Applying Principal Components Regression (PCR) on the left, respective logistic regression on the right for the `Iris` (binary) dataset (with two classes) using the first principal component only

where the $\sigma(\mathbf{w}^\top \mathbf{x}^{(i)})$ represents the ***sigmoid function*** or ***logistic function*** and its form is shown in Fig. 6.4. A sigmoid function is a mathematical function having a characteristic "S"-shaped curve or sigmoid curve, and a common example of the sigmoid function family is the logistic function defined by the formula.

$$\sigma(z) = \frac{1}{1+\exp(-z)} \tag{6.2}$$

Notice that $\sigma(z)$ for $z = \mathbf{w}^\top \mathbf{x}$ tends towards 1 as $z \to +\infty$, and tends towards 0 as $z \to -\infty$. Because the sigmoid function "squashes" any real value into the range of $[0,1]$ we may interpret the output of the logistic model $h_{\mathbf{w}}(\mathbf{x})$ as a probability. Our goal is to search for a value of $\mathbf{w}$ so that the probability $\mathbb{P}(y=1|\mathbf{x}) = h_{\mathbf{w}}(\mathbf{x})$ is large when $\mathbf{x}$ belongs to the first class and small when $\mathbf{x}$ belongs to the second class (so that $\mathbb{P}(y=0|\mathbf{x}) = h_{\mathbf{w}}(\mathbf{x})$ is large). The derivative of the sigmoid function is:

$$\frac{\mathrm{d}\sigma(z)}{\mathrm{d}z} = \frac{-\exp(-z)(-1)}{(1+\exp(-z))^2} = \frac{1}{1+\exp(-z)} \frac{1+\exp(-z)-1}{1+\exp(-z)} = \sigma(z)(1-\sigma(z)) \tag{6.3}$$

As we stated, the output of the logistic regression is a probability and, in case of binary classification, we should chose a boundary based on the probability value:

$$\hat{y}^{(i)} = \begin{cases} 1, & \text{if} \quad \mathbb{P}(y^{(i)}=1|\mathbf{x}^{(i)}) = \sigma(\mathbf{w}^\top \mathbf{x}^{(i)}) \geq 0.5 \\ 0, & \text{if} \quad \mathbb{P}(y^{(i)}=1|\mathbf{x}^{(i)}) = \sigma(\mathbf{w}^\top \mathbf{x}^{(i)}) < 0.5 \end{cases} \tag{6.4}$$

If we define the cost function $J(\mathbf{w})$ as the sum of squares error, as we did for linear regression, then we end up in a non-convex and not continuous cost function because - in case of binary classification - the error values for each observation would be either 1 or 0, depending whether we have a miss-classification or a good classification for a given $\{x^{(i)}, y^{(i)}\}$, and the sum of squares residuals for N observations would take one of the discrete values $0, 1, 2, ..., N^2$ (N^2 would be in the case if all N observations are misclassified).

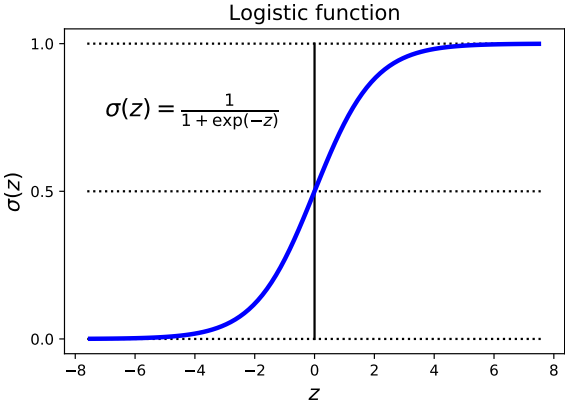


Fig. 6.4 Logistic function stretching all the values into the interval (0,1), thus the output of the logistic function can be interpreted as a provability

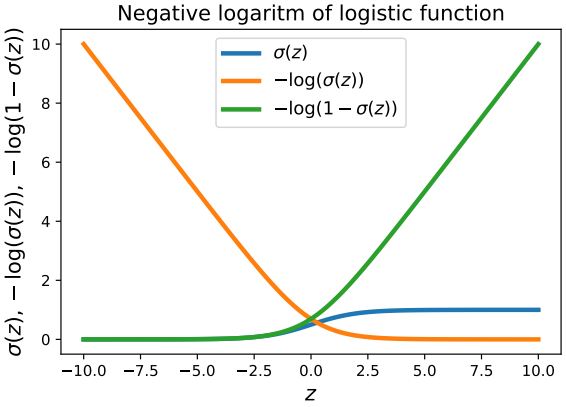


Fig. 6.5 The negative logarithm of the Logistic function, respective the negative logarithm of the one minus the logistic functions

So, given the logistic regression model, how do we fit $\mathbf{w}$? If we look at the graph of the functions plotted in Fig. 6.5 we have the following intuition: if a misclassification occurs for $y^{(i)} = 1$ with $\mathbb{P}(y^(i) = 1|\mathbf{x}^{(i)}) < 0.5$ then the negative logarithm of the probability (orange curve for $z < 0$) starts increasing rapidly and the greater the misclassification (the more the probability tends to 0) the greater the punishment on the cost function; on the other hand, if misclassification occurs for $y^{(i)} = 0$ with $\mathbb{P}(y^(i) = 1|\mathbf{x}^{(i)}) \geq 0.5$ then the negative logarithm of probability (green curve for $z > 0$) starts increasing rapidly and the greater the misclassification (the more the probability tends to 1) the greater the punishment on the cost function.

The choice between the two functions can be done using the actual value of $y^{(i)}$. With this intuition in mind, the following cost function could be created[1]:

$$J(\mathbf{w}) = -\sum_{i=1}^{N} \left(y^{(i)} \log(\sigma(\mathbf{w}^\top \mathbf{x}^{(i)})) + (1 - y^{(i)}) \log(1 - \sigma(\mathbf{w}^\top \mathbf{x}^{(i)})) \right) \tag{6.5}$$

The residuals can be defined in the same way as we did in case of linear regression, i.e. $r^{(i)} = y^{(i)} \hat{y}^{(i)}$. Then the residual vector is defined as:

$$\mathbf{r} = \begin{bmatrix} y^{(1)} - \sigma(\mathbf{w}^\top \mathbf{x}^{(1)}) \\ y^{(2)} - \sigma(\mathbf{w}^\top \mathbf{x}^{(2)}) \\ \vdots \\ y^{(N)} - \sigma(\mathbf{w}^\top \mathbf{x}^{(N)}) \end{bmatrix} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix} - \sigma \left(\begin{bmatrix} (\mathbf{x}^{(1)})^\top \mathbf{w} \\ (\mathbf{x}^{(2)})^\top \mathbf{w} \\ \\ (\mathbf{x}^{(N)})^\top \mathbf{w} \end{bmatrix} \right) = \mathbf{y} - \sigma(\mathbf{X} \cdot \mathbf{w}) \tag{6.6}$$

where $\sigma()$ represents a universal function in NumPy, and it is applied element-wise on the vector (see Appendix). The residual vector resembles with the residual vector obtained for multiple linear regression (see Eq. 3.8), except for the sigmoid function.

Now, let's try to write the cost function in a compact matrix form. Let's start with the sum of the first element withing the summation in Eq. 6.5:

[1] In Sec. 6.3.2 this cost function would be defined with the help of cross-entropy

Appendix A
scikit-learn API reference

A.1 Introduction

Scikit-learn is a simple and efficient tool for data mining and data analysis built on NumPy, SciPy, and matplotlib. It is an open-source, commercially usable (BSD license) machine learning library[1]. This appendix aims to provide a concise reference to book sections and Jupyter Notebook lab exercises of the API of the scikit-learn library.

The scikit-learn library has a well-structured API. Each machine learning algorithm within scikit-learn has a constructor with initialization parameters. Some of these parameters are the so-called hyperparameters of the model, which influence the behavior of the machine learning algorithm and the obtained results. Those parameters can be requested later on or even changed. Every machine learning algorithm also has a method for fitting the data and a method for predicting the output. If you change the parameters of the algorithm, a new fitting is required.

A.2 Common methods

Common methods in scikit-learn machine learning algorithms are:

- `get_params()`: getting the (hyper-)parameters for the machine learning estimator,
- `set_params()`: setting the (hyper-)parameters for the machine learning estimator,
- `fit(X,y)`: fitting the (training) data $\{X,y\}$ using the machine learning algorithm,
- `predict(X)`: predicting y for the (test) data X using the fitted (trained) machine learning estimator.

Some of the algorithms also have a combined `fit_predict(X)` method. With this method, one can train the machine learning algorithm on the data and also pre-

[1] For more information, see https://scikit-learn.org/stable/index.html.

dict the outcome for this data. There are also objects in scikit-learn, which only preprocess the data that other algorithms may use, so they have a `transform(X)` method instead of predict. Again, some of these objects may possess a combined `fit_transform(X)` method, so one can fit the transformation on the data and also transform the data at the same time.

Each object (machine learning algorithm) within scikit-learn also has several attributes that end in an underscore. These attributes are the result of fitting the data[2]. In the remaining part, we will list only those methods of the scikit-learn library that have been discussed in this book. References to the appropriate book sections and exercises are also provided.

A.3 Grid Search

Hyper-parameters are parameters that are not directly learned using the data but rather help in obtaining the optimal estimator for the data. A typical example would be the λ value for the ridge regression or the learning rate for α for SGD. A hyper-parameter search consists of:

- an estimator (like ridge regressor or ridge classifier);
- a (hyper-)parameter space;
- a method for searching or sampling candidates in this parameter space;
- a cross-validation scheme;
- a score function.

The score function will drive which hyper-parameter value would represent the optimal choice for our estimator.

A.4 Pipelines and Composite Estimators

In most machine learning projects, the data that you have to work with is unlikely to be in the ideal format for producing the best-performing model. There are quite often several transformational steps, such as imputation, feature scaling numerical variables or encoding categorical variables, that need to be performed. Scikit-learn has built-in functions for most of these commonly used transformations in its preprocessing package. However, in a typical machine learning workflow, you will need to apply all these transformations several times. First, you need to apply when training the model, then again to validate your model with the test data, and finally several times on new data you want to predict.

Moreover, transformers are usually combined sequentially with regressors or classifiers to build a composite estimator. Instead of writing your function for this, scikit-learn `Pipeline` is a great tool to simplify this process. For example, when using PCR, first you apply standardization and PCA whitening, then the linear regression with a reduced set of principal components. The intermediate steps of the

[2] For a complete scikit-learn API reference, please visit https://scikit-learn.org/stable/modules/classes.html.

pipeline must implement fit and transform methods, and the final estimator only needs to implement the fit method. As the name suggests, pipeline allows stacking multiple processes into a single scikit-learn estimator. The `Pipeline` class has `fit`, `predict`, and score method just like any other estimator. All estimators in a pipeline, except the last one, must be transformers (i.e., must have a transform method). The last estimator may be any type (transformer, classifier, etc.). Pipeline serves multiple purposes:

- Convenience and encapsulation: You only have to call `fit` and `predict` once on your data, thus makes your workflow much easier to read and understand;
- Joint parameter selection: You can grid search over parameters of all estimators in the pipeline at once;
- Safety: Pipelines help avoid leaking statistics from your test data into the trained model in cross-validation by ensuring that the same samples are used to train the transformers and predictors.

Many datasets contain features of different types, like numerical, ordinal, or nominal variables, where each feature type requires separate preprocessing or feature extraction steps. Often it is easiest to preprocess data before applying scikit-learn methods using other libraries like pandas. However, that might cause data leakage and does not allow the inclusion of any parameter used during preprocessing in the hyperparameter search. The `ColumnTransformer` helps to perform different transformations for different columns (features) of the data within a Pipeline that is safe from a data leakage point of view, and that can be parametrized. `FeatureUnion` on the other hand, concatenates results of multiple transformer objects. This estimator applies a list of transformer objects parallel to the same input data, then concatenates the results. So, `FeatureUnion` applies different transformers to the whole input data and then combines the results by concatenating them, while `ColumnTransformer`, on the other hand, applies different transformers to different subsets of the whole input data and again concatenates the results.

`TransformedTargetRegressor` is useful for applying a non-linear transformation to the target y in regression problems. Pipeline and feature union is implemented in `sklearn.pipeline`, while the composite estimators are implemented in `sklearn.compose` module (references to the book sections and lab exercises are listed in Table A.1).

Table A.1 Pipeline and composite estimator

Scikit-learn name	Book reference	Exercise reference
Pipeline	Sec. 3.5.12	Ex. 3.7, Ex. 3.14
FeatureUnion	Sec. 3.5.12	
ColumnTransformer	Sec. 3.5.12	Ex. 3.7, Ex. 3.14
TransformedTargetRegressor	Sec. 3.5.12	Ex. 3.14

A.5 Generalized Linear Models

The `sklearn.linear_model` module implements generalized linear models.

A.5.1 Linear Regression Models

The linear models discussed in the book placed in Table A.2 but not listed in alphabetical order. Before using any of the linear models of scikit-learn, you should import either the whole module or the appropriate model object only.

Table A.2 Generalized Linear Models

Scikit-learn name	Book reference	Exercise reference
LinearRegression	Sec. 2.6 for SLR, Sec. 3.3 for MLR	Ex. 2.7, Ex. 3.1 - Ex. 3.4
Ridge RidgeCV Lasso LassoCV ElasticNet ElasticNetCV		
Lars LarsCV LassoLars LassoLarsCV		
HuberRegressor BayesianRidge SGDRegressor		

A.5.2 Regression Metrics

The `sklearn.metrics` module implements some loss, score, and utility functions to measure regression performance. Those discussed in the book listed in Table A.3.

Table A.3 Regression metrics

Scikit-learn name	Book reference	Exercise reference
max_error		
explained_variance_score	Sec. **??**	Ex. **??**
r2_score	Sec. 2.6.2	Ex. 2.7
mean_squared_error	Sec. 2.6.2	Ex. 2.7
mean_squared_log_error	Sec. **??**	Ex. **??**
mean_absolute_error	Sec. 2.8.1	Ex. 2.6
median_absolute_error	Sec. 2.8.2	Ex. 2.6

Appendix B
Brief Introduction to NumPy

B.1 Introduction

NumPy, short for Numerical Python, is the fundamental package required for high-performance scientific computing and data analysis. This appendix aims to provide a quick introduction to the NumPy library and the corresponding mathematical notations used throughout the book. However, this appendix is not a full description of NumPy, nor it tells you how to install it[1]. Instead, this appendix is focusing only on those parts, which are intensively used in the accompanying Jupyter Notebooks. Moreover, the appendix helps you to understand how different mathematical formulas can be converted easily to NumPy code directly.

Before using the NumPy library, you shall import to your environment with the following command[2]:

```
>>> import numpy as np # Import library with 'np' alias
```

The reason for using the NumPy library in the accompanying Jupyter Notebooks is the ease of use, less coding, and higher execution speed. Moreover, the mathematical formulas derived in the book can be implemented directly in a single line of code. Thus, the implementation of a machine learning algorithm can be done easily in a few lines of Python code, which helps you better understand the implementation of a machine learning algorithm.

Note *This appendix may also help you refreshing your linear algebra knowledge. In case you lack that know-how, checking the videos in the footnotes might provide some very basic hints about this broad topic.*

[1] You can find a full reference to Numpy at https://docs.scipy.org/doc/numpy/

[2] Code lists within this appendix are obtained with the old iPython console. Depending on your environment the prompt might be different.

B.2 Basic Data Structure of NumPy

One of the key features of NumPy is the *N*-dimensional array object. NumPy's array class is called `ndarray`, which is a fast, flexible container for large homogeneous data. It is an array of elements (usually numbers but can be any Python object), all of the same type, indexed by a tuple of positive integers. These arrays enable you to perform mathematical operations on whole blocks of data using similar syntax to the equivalent operations between scalar elements. Thus, no `for` loops are required, which leads to fewer lines of code and much faster execution.

In NumPy, the number of dimensions of an array is called `rank`, while the `axis` refers to a single dimension of a multidimensional array. Thus, the dimension of the multidimensional arrays also defines how many axes are present in the array. The `shape` of an array tells you the size of each axis, that is, how many elements are in each dimension. In Fig. B.1, a multidimensional array is visualized.

All `ndarrays` are homogenous: every item takes up the same size block of memory, and all blocks are interpreted in the same way. How each item in the ar-

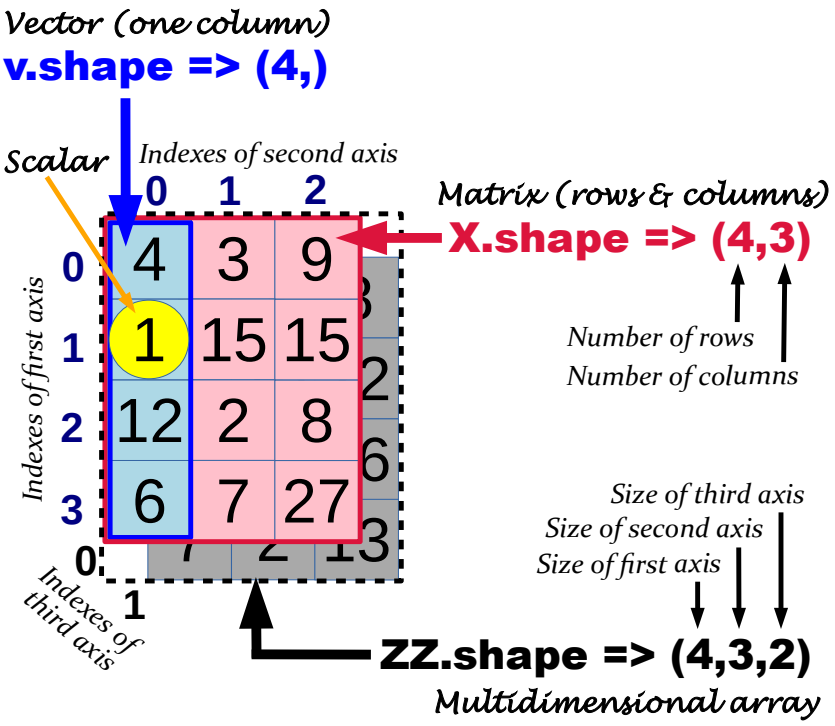


Fig. B.1 The structure of a multidimensional array and the distinction between scalar, vector, matrix, and the generalized multidimensional array

ray is to be interpreted is specified by a separate data-type object, as an instance of `dtype` class. An instance of class `ndarray` consists of a contiguous one-dimensional segment of computer memory, combined with an indexing scheme. The ranges in which the indices can vary is specified by the `shape` of the array. Because each data type requires a different storage size, the `dtype` tells you how much bytes the underlying code should skip accessing the next element in the memory (see Fig. B.2).

Computer memory is inherently 1-dimensional, and there are many different schemes for arranging the items of an N-dimensional array in a 1-dimensional block. NumPy is flexible, and `ndarray` objects can accommodate any stridden indexing scheme but the default is the row-major order (used in C) when the rightmost index "varies the fastest"[3]. Strides represent the number of bytes to jump to find the next element, and one stride per dimension is defined. For the rightmost index, the stride is equal to the data size.

```
>>> x=np.array(1) # Assing to 'x' the array with 1
>>> x             # Printing out the value of the array
1
>>> x.size        # Number of elements in the array
1
>>> x.itemsize    # Number of bytes used by an element
4
>>> x.dtype       # Type of the data in the array
dtype('int32')
```

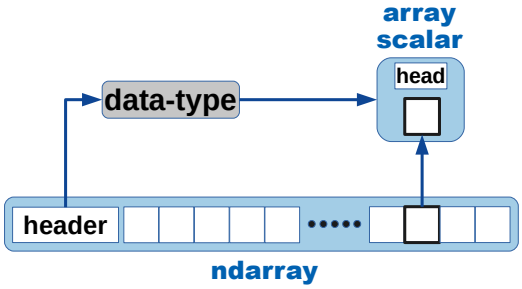


Fig. B.2 Conceptual diagram of `ndarray` with three fundamental objects: 1) the `ndarray` itself, 2) the data-type object describing the layout of a single fixed-size element of the array, 3) the array-scalar Python object that is returned when a single element of the array is accessed

[3] Memory layout can affect performance because CPU pulls data from main memory to its cache in blocks. If many array items (consecutively operated on) fit in a single block (small stride), then fewer transfers are needed from memory, thus faster execution. `numexpr` is designed to mitigate cache effects when evaluating array expressions (see at https://numexpr.readthedocs.io/projects/NumExpr3/en/latest/), while `numba` is a compiler for Python code, that is aware of NumPy arrays (see at https://numba.pydata.org/).

Note *Remember, when binding a variable to an object in Python means that we are setting a name to hold a reference to an object. Thus, assignments create references and not copies, and as such, variable names in Python do not have any intrinsic type - only the objects have - as they are holding only memory addresses of the referenced object. Based on the data object assigned to the variable name, Python determines automatically the type of the reference. A reference is deleted via garbage collection after any variable names bound to it have passed out of scope.*

A scalar in NumPy terminology is a `ndarray` with rank 0 (or dimension 0), also called array scalar. In NumPy, there are 24 new fundamental Python types to describe different types of scalars, and these are mostly based on the types available in the C language[4]. Depending on the `dtype` of the NumPy array, the element of the 0-dimensional array can be an integer, a float, a character, or even any Python object or even structured data type[5]. Zero dimensional arrays can be converted to scalars directly (i.e., they can be cast to normal Python objects).

A vector in NumPy terminology is a `ndarray` with rank 1 (or dimension 1). It has a single axis containing a series of objects (numbers). Elements within the vector can be accessed by indexing, and the first element has index 0. When accessing a single element within a vector, the result is a scalar. Vectors can be used to store the (Euclidean) coordinates of an object, or the observations connected to a feature, or a single observation with multiple features, etc. A vector can be converted to a scalar only if it contains a single element.

A matrix in NumPy terminology is a `ndarray` with rank 2 (or dimension 2). It has two axes, the first axis corresponding to the number of rows, while the second axis to the number of columns. Matrices can be used to store gray images or several observations, each with multiple features, or multiple observations through time, etc. When indexing an element of the matrix, the first index always identifies the row number, while the second index the column number. Be aware that indexing always starts from zero, so if you want to access the element highlighted with yellow in Fig. B.1, the first index is 1, while the second index is 0. A matrix can be converted to a scalar only if it contains a single element.

A multidimensional array of rank 3 (or dimension 3) can be seen as a collection of tables, each with rows and columns. Such arrays can be used to store color images, where each 2-dimensional table is assigned to a base color. Or to store panel data, data that tracks attributes of a cohort (group) of individuals over time. For example, in medical research, some subjects are given a placebo, while others are given an active drug, and the two groups are compared, based on repeated measures of data, like blood pressure, viral load, immune status, etc. Thus, each table is associated with a patient, with rows containing measured data at each follow-up time. Looking at Fig. B.1, the last index would represent the patient, the first index the follow-up time, while the second index the attributes related to the physiological status of the

[4] For a complete list of NumPy scalars check https://docs.scipy.org/doc/numpy/reference/arrays.scalars.html

[5] Structured data types are formed by creating a data type whose field contains other data types.

patient. But of course, nothing prevents you to change the order of the axis, so the first axis would be associated with the patients.

Higher-dimensional arrays can be tougher to picture, but they will still follow this "arrays within an array" pattern. For example, a video stream without compression would require a 4-dimensional array for storing it, and the four axes would be: image number, height, width, and color channel.

In what follows, we focus on some peculiarities of scalars, vectors, and matrices, and in general, what operation or array manipulations can be done using NumPy. However, this appendix shall be seen only as a quick guide for those who are not familiar with NumPy but have some basic knowledge in programming, not necessarily in Python. This appendix also helps in understanding how the multidimensional arrays and the operations performed on them are linked to mathematical symbols used in the book.

B.3 Scalars

A scalar, denoted by small italic letters in the book, like $a, b, c, ..., x, y, z, ...$, represents a single value. A scalar variable in Python points to a memory cell storing a single value (like integer or real number, or any other object). In the Jupyter Notebooks accompanying the book, only small letters are used in Python for variables names (exception is the number of observations N, the number of features D, and the number of classes K).

```
>>> x=1 # Assign to 'x' the constant value of 1
>>> x
1
```

In NumPy terminology, a scalar represents a `ndarray` of rank 0 (or dimension 0). We can convert a Python object to a NumPy array. The rank of the array can be found with the help of the `ndim` attribute, while the size of each axis with the `shape` attribute, the latter being a tuple: An array with rank 0 does not have any axis, so its shape will be an empty tuple.

```
>>> x=np.array(1) # Convert the scalar to array
>>> x             # and show the result
array(1)
>>> x.ndim        # Check its dimension
0
>>> x.shape       # Check its shape
()                # no shape for array scalar
>>> x=int(x)      # Cast 'x' to integer
>>> x             # and find out its value
1
```

B.4 Vectors

A vector, denoted by small bold letters in the book, like $\mathbf{a},\mathbf{b},\mathbf{c},...,\mathbf{x},\mathbf{y},\mathbf{z},...$, represents an array containing multiple scalar values ordered in a single column. The number of values stored in a vector is denoted by big italic letter (like $D,...,N,M,...$) and the elements of the vector are identified by subscript (or superscript) indexes:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \tag{B.1}$$

Note *By mathematical definition, a vector always represents a column vector having N rows and one column. A vector is characterized by a magnitude and a direction, and it can be imagined as an arrow, starting from the origin of the Euclidean space. As a computer scientist, you may think of a vector as on the ordered list of numbers, where the numbers represent the coordinates of the tip of the vector in a N dimensional hyperspace*[6]*. In NumPy terminology, a vector might be represented by a* `ndarray` *of rank 1. However,* `ndarray` *of rank 1 can represent both a column vector and a row vector, which leads to unexpected results when performing operations between vectors, or between a vector and a matrix. For this reason, in the lab exercises, a vector is always represented by rank 2* `ndarray` *similar to matrices. Thus, in lab exercises, the only difference between a vector and a matrix is that a vector will always have only one column, i.e., the size of the second axis will always be equal to 1.*

In the Jupyter Notebooks accompanying the book, vector variables are denoted by small letters ending in an underscore. Since there are no bold letters in the code, this underscore at the end will distinguish vectors from scalars. The underscore also might resemble the small arrow placed above the vector variables in physics.

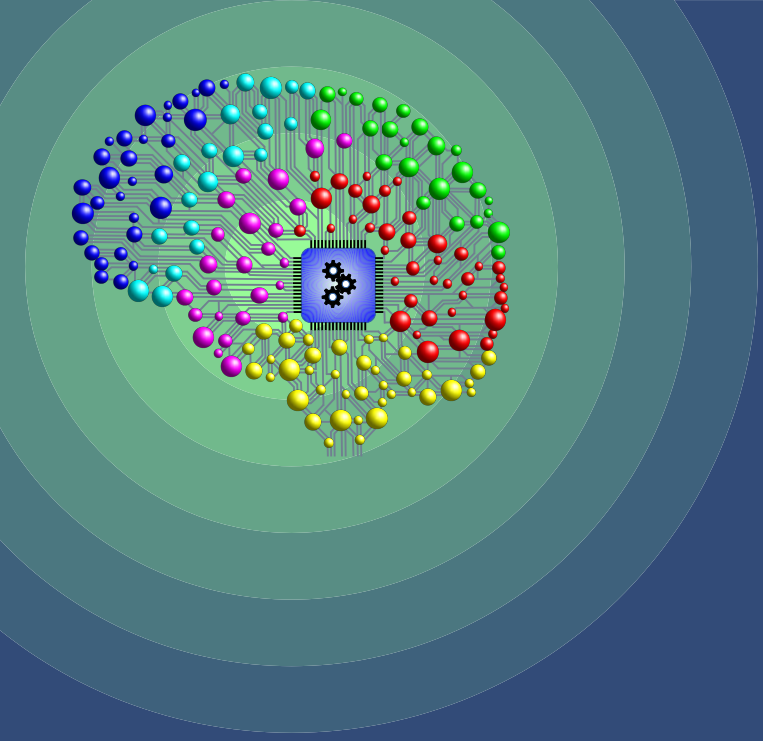
B.4.1 Vector Initialization

There are several ways to create and initialize a vector in NumPy. The simplest way is to initialize a vector to have all their values set to either zero or one using `zeros`, respective `ones` methods. The third option is to provide a number to fill the vector elements with `full` method.

```
>>> x_=np.zeros(3)  # Create x_ of size 3 with zeros
>>> x_
array([0, 0, 0])
>>> x_=np.ones(5)   # Create x_ of size 5 with ones
```

[6] If you are not familiar with the notion of vector and what it represents geometrically, please check the video from 3Blue1Brown at https://youtu.be/fNk_zzaMoSs

Index



Ferenc Farkas, has a Ph.D. in Electrical Engineering and Master of Arts in Economics, with more than 25 years of experience in SW and HW development, currently working as a system architect at an international company.