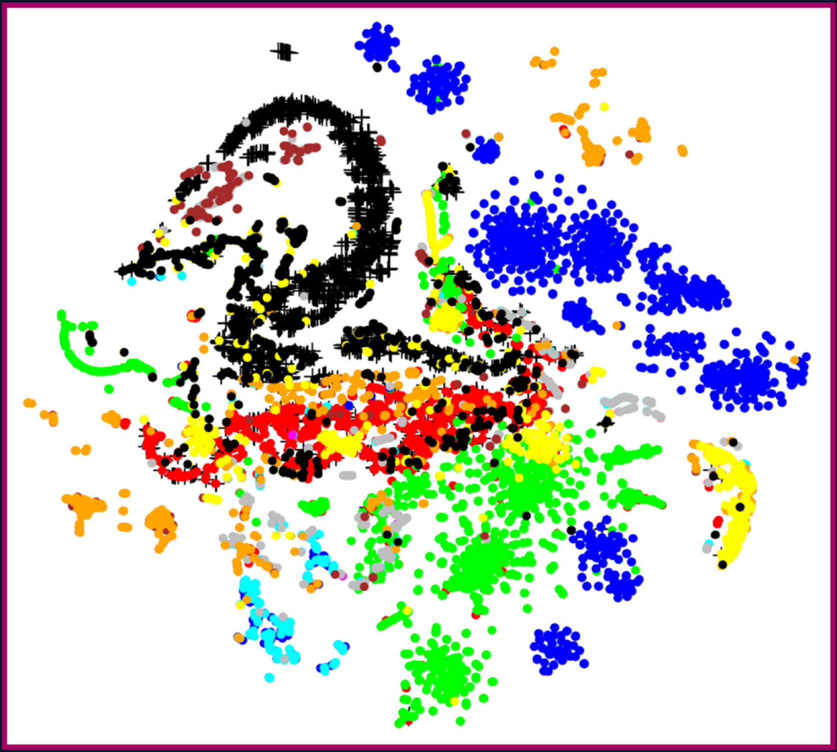
DATA SCIENCE SERIES

CYBERSECURITY ANALYTICS



RAKESH M. VERMA
DAVID J. MARCHETTE

CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

Cybersecurity Analytics

Cybersecurity Analytics

Rakesh M. Verma
University of Houston, Houston, TX
David J. Marchette
Naval Surface Warfare Center, Dahlgren, VA

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business
A CHAPMAN & HALL BOOK

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Printed on acid-free paper

International Standard Book Number-13: 978-0-367-34601-0 (Hardback)
978-0-367-34602-7 (Paperback)

Library of Congress Cataloging-in-Publication Data

Names: Verma, Rakesh M., author. | Marchette, David J., author.
Title: Cybersecurity analytics / Rakesh M. Verma and David Marchette.
Description: Boca Raton, FL : CRC Press, 2020. | Series: Chapman & Hall/CRC data science series | Includes bibliographical references and indexes. |
Summary: “This book organizes in one place the mathematics, probability, statistics and machine learning information that is required for a practitioner of cybersecurity analytics, as well as the basics of cybersecurity needed for a practitioner”-- Provided by publisher.
Identifiers: LCCN 2019029587 (print) | LCCN 2019029588 (ebook) | ISBN 9780367346010 (hardback) | ISBN 9780367346027 (paperback) | ISBN 9780429326813 (pdf)
Subjects: LCSH: Computer security.
Classification: LCC QA76.9.A25 V456 2020 (print) | LCC QA76.9.A25 (ebook) | DDC 005.8--dc23
LC record available at https://lccn.loc.gov/2019029587
LC ebook record available at https://lccn.loc.gov/2019029588

In loving memory of my mother, Usharani Verma

Rakesh M. Verma

To my wife, Susan

David J. Marchette

Contents

Preface

This book arose from a panel discussion [71] at the ACM International Workshop on Security and Privacy Analytics (2015) [449] that led to the paper, [455]. Part of the impetus for this book also came from the Security Analytics course that the author, Rakesh M. Verma, has been teaching at the University of Houston since 2015.

It is an attempt to organize in one place the mathematics, probability, statistics and machine learning information that is required for a practitioner of cybersecurity analytics, as well as the basics of cybersecurity needed for a practitioner. We feel that any collaboration between a data scientist and a cybersecurity expert cannot be fully productive, unless there is some shared vocabulary and mutual understanding, which is our objective in this book. We aim to provide the cybersecurity analyst with the tools to understand the mathematics and machine learning methods that are used for most cybersecurity analytics.

Such an undertaking cannot be comprehensive; however, we aim to give some background on the main areas of knowledge necessary for cybersecurity analytics, and provide a basic understanding of the ideas, with citations for further reference. It is assumed that a reader has a basic understanding of mathematics at the college level. We attempt to give enough background to understand the examples and applications to cybersecurity, and provide references that can fill in the gaps. We intend the book to provide cybersecurity analysts an introduction to some of the advanced mathematics that applies to the problem domain.

In *The Art of War* ([442]), it is said:

If you know the enemy and know yourself, you need not fear the result of a hundred battles. If you know yourself but not the enemy, for every victory gained you will also suffer a defeat. If you know neither the enemy nor yourself, you will succumb in every battle.

The main purpose of the techniques discussed in this book is to provide methods to both "know thyself" and "know thy enemy". The former involves methods to understand the environment in which the systems are utilized, the data that one observes under normal conditions, and the ways to protect the systems from attack. The latter involves understanding what attacks "look like" when they are mounted against the system, design methods to detect attacks, and methods to detect unusual events that may correspond to novel, previously unseen attacks. While one also wishes to "know thy enemy" in the sense of knowing who the attackers are and what their motivations and capabilities are, this is mostly beyond the scope of this book, and is touched on only briefly in Chapter 3.

The book is organized as follows. The introduction gives some insight into our philosophy and some of the main ideas we wish the reader to take away from the book. The chapter on data anaytics outlines the basic techniques of data processing and data analysis necessary for cybersecurity applications. It describes many of the important issues and gives an introduction to some of the important concepts. Basic cybersecurity concepts are introduced. The basic mechanisms, threats and challenges are described from a data analytics point of view.

The chapter on probability and statistics gives an overview of the key concepts, and provides some discussion of practical applications. This is followed by a chapter on data mining. In some circles this term has become passé, and replaced by data analytics or data science. In this book it is used to refer to the unsupervised aspects of machine learning, including clustering, and anomaly detection, but also association rules – "if you liked this movie, you might like these" – and manifold learning, which is a method for reducing the dimensionality of a data set by modeling the "manifold structure" of the data.

The data mining chapter is followed by a chapter on machine learning, which in this book refers primarily to supervised classification. The distinction between supervised and unsupervised methodologies is important, which is why the chapters are split, but the naming convention of data mining versus machine learning for the two approaches is nonstandard. The machine learning also contains a section on topological data analysis, which is an unsupervised method for finding structure in data that is related to manifold learning, and is becoming more and more a standard tool in machine learning. Also, we give a very brief introduction to neural networks in this chapter, which have both supervised and unsupervised variants.

The next two chapters apply the previous methods to the analysis of textual data. Examples of these are documents, emails, log files, etc. Finally, the book ends with a short chapter on big data techniques and processing at scale.

There are appendices in which the basic linear algebra and graph theory required by this book are discussed. The book also has a fairly extensive Bibliography, and an effort has been made to provide references to a wide range of papers and books relevant to the field, as well as background material and some material of historical interest.

In writing a book, there is always the question of what material to present, and what areas to leave undiscussed. This is a very personal decision, based on the preferences, expertise, and interests of the authors, and the level of experience and knowledge expected of the reader. There are many things missing from this book that another author might choose to present. There are also several choices that were made, consciously and subconsciously, on how to present the material and the depth and detail of the presentations.

Some of the organization of the book was dictated by the decision to write a book jointly, and to set an ambitious deadline for the book, and some of the organization decisions were made when we wrote the paper [455]. The deadline forced us to devote the necessary time, even when other constraints on our time pressed us for attention. The fact that we were geographically dispersed was mitigated through the use of Overleaf,[1] which was considerable help in working on this project and keeping things organized.

The deadline also helped to scope down some of our ambitions. There are clearly topics that we would like to devote more discussion to. There are more examples we would like to provide. There are topics that we would like to include. We hope that the decisions on what to include and exclude is mitigated somewhat by the Bibliography, and the level of detail and examples provided is sufficient to get the important points across and give the readers the tools to explore further, and apply the techniques to their data and problems.

A book of this scope and length is a major undertaking, and would not be possible without significant help and sacrifices. With deep gratitude and pleasure, the authors credit the following, who generously gave up their time and/or help.

[1]https://www.overleaf.com/

Acknowledgments

Rakesh Verma thanks his family, Rashmi, Sudha and Sneha, his mentors, colleagues, friends, and his students in the Reasoning and Data Analytics for Security (ReDAS) Laboratory: Tanmay Thakur, Nirmala Rai, Vasanthi Vuppuluri, Keith Dyer, An Nguyen, Arthur Dunbar, Boris Chernis, Luis F.T. Moraes, Shahryar Baki, Daniel Lee, Ayman El Aassal, Avisha Das, Vibhu Sharma, Devin Crane, Marc Magnusson, Houtan Faridi, Uroosa Ali, and Nabil Hossain, without whom none of this would have been possible. Thanks to Narasimha Shashidhar Karpoor for early discussions on phishing. Thanks also to the National Science Foundation and the Department of Defense (Army Research Laboratory) for supporting some of the research described in this book.

David Marchette thanks his family, Susan, Steven, Jeffrey and Katy, and his mentors, colleagues and friends, without whom none of this would have been possible.

Both authors would like to thank the following colleagues and students for generously contributing their time and reviewing parts of the manuscript: Ricardo Vilalta for a close reading of the Machine Learning chapter, Wenyaw Chan and Luis Moraes for feedback on the Statistics chapter, Chester Rebeiro for thoughtful comments on the Basics of Security and Security Analytics chapter, Edgar Gabriel for a careful reading and good feedback on the Big Data for Security chapter, Arjun Mukherjee for constructive comments on the Text Mining chapter, Carlos Ordonez and Daniel Lee for reviewing the Data Mining chapter, and Avisha Das for excellent suggestions on the NLP chapter. Of course, we are responsible for all remaining errors and omissions.

Chapter 1

Introduction

This book is an attempt to organize in one place the mathematics, probability, statistics, machine learning, and text processing information that is required for a practitioner of cybersecurity analytics, as well as the basics of cybersecurity needed for a practitioner. Such an undertaking cannot be comprehensive; however, we aim to give some background on the main areas of knowledge necessary for cybersecurity analytics, and provide a basic understanding of the ideas, with references for further study.

There are many books and articles relevant to this topic. To mention just a few, see for example [299, 333, 402, 404, 445]. See also the computer security section of your local bookstore. See [62] for a survey of machine learning methods for security. There are also several books aimed specifically at data analysis and machine learning for cybersecurity. See for example [3, 82, 293]. These are compendiums of papers or chapters written by experts in the field, and provide a number of specialized investigations in areas touched on in this book. For a more introductory text, see [208]. There are also a number of articles that survey machine learning methods for security. See for example [62].

Most of cybersecurity is focused on defense – firewalls, secure coding principles, securing computers through a Security Technical Implementation Guide (STIG), virus scanners, phishing and spam detectors, and training users in best practices. Organizations will often hire a red team, a group of security experts who analyze their network/computers for vulnerabilities and errors, to allow the company to better harden the systems. Others keep security experts on staff, or contract with a security company to provide the information technology infrastructure for them.

Some of this is enhanced through statistical and machine learning algorithms, and it is these algorithms that are the focus of this book. Basic design flaws in software such as buffer overflow vulnerabilities are best addressed at the source, through better coding practices. However, we do not live in a perfect world, mistakes are made, and most software is in binary format without access to the source code. Thus, methods need to be developed to detect vulnerabilities without access to the software, and often this means one must observe the actions of software, network traffic, or user activity to detect "unusual" events, or known attacks. This is where pattern recognition techniques, such as statistical and machine learning methods, come in.

The news is full of the terms Artificial Intelligence (AI) and Machine Learning (ML). Artificial intelligence refers to writing code to mimic intelligent behavior. Code that guides a robot through a maze, chess playing software, and recommendation systems in online commerce may all be examples of artificial intelligence. Note that the first example may be purely algorithmic – there are many path-following algorithms that have been developed – or it may be more sophisticated if for example there are potential obstacles that must be detected and avoided. The early chess-playing machines utilized powerful search methodologies that allowed them to "look ahead" a number of moves, as well as pruning algorithms to attempt to keep the search tree to a manageable size. These were also algorithmic; modern methods have at least some aspect of learning in them – they learn from playing, although even these algorithms will often maintain some core of "look ahead" searching in them.

A key component of artificial intelligence is search – the ability to rapidly find matches to a pattern in a large set of data. In fact, it has been said that *artificial intelligence is search.*[1] An interesting book on the history of artificial intelligence is [306], in which Alan Turing is quoted (page 70) as supporting this sentiment. Clearly AI is more than just better search algorithms, but much of AI can be understood in terms of "searching for patterns" and AI algorithms often contain some aspect of search.

The field of AI encompasses the field of machine learning as a subfield. The distinction is that machine learning utilizes data to "learn" the important patterns or actions, rather than requiring a clever person to invent an algorithm and implement it in code. The machine learns the desired capability through processing *training data* which is representative of the task at hand. Obviously, this requires clever people to come up with training algorithms to allow this to happen, and this book will describe a number of these and illustrate their application for several cybersecurity purposes.

Of course, as noted above, there's more to AI than search, and there's more to ML than learning from data. This book focuses on techniques related to data, and to the types of problems that can be addressed through data analysis and machine learning methodologies. Interested readers should investigate one of the many books written specifically on AI, machine learning, or related subjects, such as [128, 213, 371]. Readers interested in neural networks, which is touched on only briefly in this book, might find [172] of interest.

The idea of a *model* is central to the aspects of AI and machine learning that we are focused on in this book. The model provides a representation of the data that allows one to make predictions about new data. In machine learning these models are statistical in nature, and the parameters of the model (and in some cases the model form itself) are determined from the data. AI methods that utilize rules also have a statistical basis to them, whether the rules are extracted from experts or learned from data. There's always uncertainty involved in any system that interacts with the "real world" and this means that one must incorporate statistical models in the solution.

In essence, one posits a family of models from which the data was drawn or which can adequately approximate the true model and defines an algorithm or set of algorithms to select the appropriate model from the family and fit the parameters of that model. It is often the case in machine learning that the family of models is not made explicit by the algorithm. For example, the random forest algorithm (Section 6.5.1) builds a large number of "decision trees" and provides a vote from these trees. While it is possible to describe the family of models that are encompassed by this algorithm – it is the family of "forests of decision trees" – only theoreticians are likely to care about the details. Similarly, feed forward neural networks (Section 6.8) are a family of models – the hidden layer structure and activation functions determining the model within the family and the weights and biases determining the parameters. In both of these examples, one does not think of the models as *generative* – that is that the data were generated from a model of this type – but rather as approximations that are sufficiently accurate to perform the desired inference.

Several data sets are used throughout the book. Some of these have known problems, and the data are used solely to illustrate the various techniques. One of the data sets, the KDD-Cup data, was constructed from the DARPA 1999 data set, which is an artificial data set originally constructed to act as a testbed for computer intrusion detection algorithms.

[1]We do not know the origin of this phrase, although there are a number of examples of it in literature. See for example https://clsimplex.com/asset/artificial-intelligence-introduction.

The original set had known problems (see [291]), and although the KDD-Cup data does not suffer from all of these (see [424]), it is old and not ideal for serious computer security work. With that said, it is a useful data set for illustrating some of the issues and techniques in machine learning, and its wide availability means that it is relatively easy for the interested reader to obtain it and test out the methodologies in this book for herself.

A tool that appears in many places in this book, in some cases prior to it being formally introduced in Chapter 6, is the nearest neighbor classifier. This is one of the most easily understood and implemented algorithms in machine learning, and is a useful tool for illustrating some of the other ideas throughout the book. The algorithm is as follows. Consider the case of classifying an email as spam or not. We are given data $\{x_1, x_2, \ldots, x_n\}$ (emails) for which one has known class labels $\{y_1, y_2, \ldots, y_n\}$ (spam or legitimate – *ham*), and a new observation (email) x which one wishes to classify. One finds the x_i closest to x and assigns class y_i to x. This is trivially generalized to produce a "vote" amongst the k closest x_i. We will use this algorithm in several places to illustrate how one method for analyzing data is/is not better than another for various purposes.

There are a number of important lessons that we wish the reader to take from this book, beyond the basic material and methodologies of analytics for cybersecurity. These are:

1. Without (good) data, there is no knowledge. This is particularly important in cybersecurity. Although a mathematician can start with a set of axioms and develop a theory that follows logically from these axioms, in the real world the axioms are mostly unknown – in spite of centuries of investigation – and are only partly and imperfectly known when we come across them. Thus, one needs data to understand the actual properties and variability of any system one wishes to analyze, and one needs data to determine what types of things one can expect to see. *A corollary to this maxim is that the quality of the data, and of the ground truth, is extremely important.*

2. As Box has said in several places and several different ways ([50, 51]): all models are wrong; some are useful. We will discuss parametric and nonparametric models in this book, and suggest that practitioners investigate different models and examples of both parametric and nonparametric methods when analyzing data or designing algorithms. It is simply not possible to find "the one true model" for anything but the most simplified and constrained situation; however, one can learn a great deal from models that are approximately correct.

3. Simple models are to be preferred to more complex models, except when they aren't. We refer often to David Hand's paper [185] in which he argues this point quite succinctly. This is also a consequence of the bias-variance trade-off: the more complex the model, the more likely it is to overfit the data. While the more complicated model can provide a better fit to the true model, assuming the extra complexity is warranted, it can only do so if there is sufficient data to accurately fit the parameters. Quoting William of Ockham: *"Entia non sunt multiplicanda praeter necessitatem"*.[2]

4. Dr. Hand also notes that the data used to train the system is very often not drawn from the same distribution as the data the system will see once it is deployed. This is particularly important in cybersecurity: the data is nonstationary, meaning that it changes in time. Some of this change is due to growth in usage, some is due to new usage such as novel applications and functionality, and some is due to the adversaries, who continue to invent new ways to compromise systems. Yet, even if the data were stationary, it would still be impossible to collect data that is representative; each

[2]"More things should not be used than are necessary," from Wikipedia, https://simple.wikipedia.org/wiki/Occam's_razor, accessed 5/27/2018.

network is different, each computer is used in a slightly different way and has slightly different applications and vulnerabilities, each organization has different policies and concerns. Thus, any algorithm developed for cybersecurity must be robust to the fact that the data that was used to train it is not fully representative of the data it will see when deployed.

5. The above point emphasizes an extremely important issue: evaluating one's solution on new data is critical. In a cybersecurity application, one must be sure that the data used to test the system is representative of the data the system will see. One must be constantly vigilant to changes in the data and changes in the environment that generates the data. This means that adaptive methods that can adjust to new situations may be attractive; care must be taken, though, to ensure that this does not introduce a vulnerability – the ability of an attacker to craft data that causes the system to adapt away from the correct model. See the references on adversarial methods in Chapter 6, and the references therein. We discuss this briefly below as well.

6. There is almost never just one correct answer to any problem. Different methods have different strengths and weaknesses, and knowing these is important to understanding which methods are most appropriate for a given task. On the other hand, utilizing several different methods can provide one with additional information about the problem, and can allow for a hedge against incorrect assumptions. With that said, there has been quite a bit of work done to investigate the performance of various algorithms against various data sets. Although there is no one universal answer, there are often guidelines that can be obtained by reading this literature. For example, in the area of classification (supervised learning), an extensive evaluation of existing methods ([150]) shows that there are a few algorithms that are consistently good: ensemble methods like random forests, and kernel methods such as support vector machines.

7. The more you understand the underlying mechanism that produces the data, the better you can design an algorithm to solve the problem you wish to solve. Similarly, the more you understand the underlying theory and applicability of a given method, the better you can apply that method in a rigorous manner to new problems.

8. The flip side of the above point is that it is important to understand the data you observe, and this requires tools to visualize and plot the data. There are a number of techniques discussed in this book, and there are many other very good books on visualization and graphics applied to data analysis. A good place to start are the books by Edward Tufte, [437, 438, 439]. See also [96, 514, 515]. The grammar of graphics ([488]) lays out the basic structure of graphics in a coherent manner, and provides insight into how the pieces of a good graphic fit together and are related. This has had a revolutionary effect on statistical graphics.

The issue of parsimony of model, or overfitting of the data, is illustrated in Figure 1.1. To illustrate the problem of choosing a model that is too complex, we fit a degree 20 polynomial to the data. Note that the polynomial interpolates through all the data points, which is clearly not desirable here. In effect, the polynomial is "memorizing" the data. Note further that outside of the data the model can be wildly different than the correct value (indicated by the solid line), and even wildly different from the nearby data points. The range of values attained by the polynomial fit in this region is $[-4279576, 6886]$, hence between the nearly vertical lines in the figure the curve is massively wrong.

To illustrate that the problem isn't just the number of parameters, we show as a dashed curve a nonparametric estimate ([161]). This uses a window to compute a locally linear fit

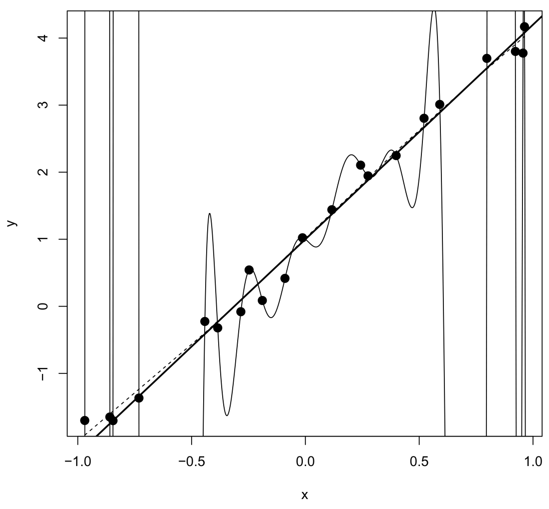


Figure 1.1: A collection of 21 points, drawn from the model denoted by the solid line with added noise. A polynomial fit of degree 20 is shown as a solid curve. A nonparametric smoother is shown as a dashed line. The vertical lines on the ends of the interval are indicative of extremely large values for the polynomial fit in these regions.

to the data, and although in a sense the number of parameters is large – in fact it is equal to the number of observations times 2 – the overall model does a good job of fitting the model for this data set.

The worst overfitting occurs near the "edge" of the data, while there is much less variability in the interior of the support of the data. However, even there, the fit between points is quite far off the correct model.

The polynomial fit in Figure 1.1 is a good one to keep in mind, particularly in light of Figure 1.2.[3] In the polynomial case, the model family is the set of all degree 20 polynomials on one variable with coefficients in $\mathbb{R}$ (or that subset of $\mathbb{R}$ that can be represented in 64-bit arithmetic). For this example, suppose our family requires that the coefficient of x^{20} be nonzero, say at least some $\epsilon > 0$ in absolute value. While the true model – a line – is not in the family, it is possible to get quite close to the model – at least in the range of the data: $[-1, 1]$ – by setting most of the other coefficients to zero, keeping only the dominant and linear terms. So the extrinsic error is quite small in this case, but one would require a very large number of observations to drive the intrinsic error to something manageable.[4]

The error between points, in regions for which there are no observations, is of concern – it corresponds to how well the algorithm "generalizes" outside of the data used to fit it. This is one of the most important aspects of machine learning: to understand the methods and the models they produce, so that one is confident that the system will operate correctly on new data. There is a new field called "adversarial learning" in which one utilizes an adversary to force a machine learning algorithm to create models that are robust to attacks. This is also used to defeat algorithms, and to highlight their inadequacies. See [173, 283, 340, 372, 420] for some examples. The more complex the model, the more difficult it may be to validate

[3]This figure is inspired by Figure 12.1 of [115].

[4]We are assuming that the error is only computed within the range of the data, since the polynomial will go to infinity at a very high rate outside of this range.

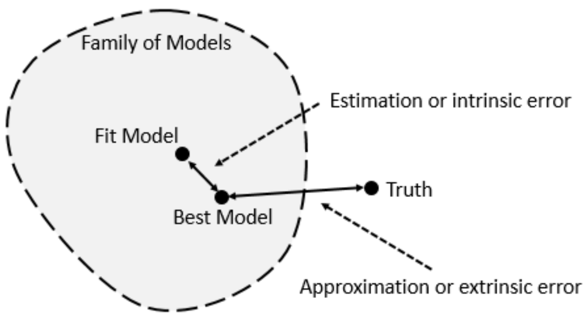

Family of Models
Estimation or intrinsic error
Fit Model
Truth
Best Model
Approximation or extrinsic error

Figure 1.2: The different errors that occur when fitting a model to data. The family of models, whether parametric or nonparametric, is given by the gray region. This contains the model which best corresponds to the true distribution, which (see the Box quote above) is not contained in the region. This results in an error that cannot be reduced, without changing the model family. Additionally, the actual model fit to a data set will not be the best possible model, and so there is error introduced there as well.

it and determine exactly "what it is doing" – and, most importantly, what it will do with new data. Once again, we come back to the lesson that we want the simplest model that performs the desired task.

Algorithms that continue to adapt – for example online learning algorithms discussed in Section (6.12) – also need to be checked to ensure they haven't "gone off the rails". A sufficiently knowledgeable adversary may be able to inject data into the system that causes it to "learn" to ignore the attack that is ultimately mounted by the adversary. At a minimum, the system needs to be checked to ensure that it still correctly handles the elements of the original data on which it was designed and trained that are still relevant. This is necessary, but not sufficient, and so these systems must be constantly monitored to ensure that they are still performing at the desired accuracy.

In [58] two "cultures" of data analytics or data science are described, which can be thought of as modeling versus algorithms. Essentially, the distinction can be thought of as between those who are focused primarily on obtaining good models, for the purpose in part of better understanding the processes that generated the data, and those who are more interested in the bottom line of performance of the algorithm. One is tempted to think of these as the scientists and the engineers, or academics and practitioners, although these analogies are imperfect. See [123] for an updated discussion of this paper and the issues it brings up.

Both modeling and algorithm performance are key to implementing successful data analytics and machine learning to solve problems in cybersecurity. However, in this book we will tend to err more on the side of the bottom line. Although the models are key to developing and understanding the algorithms, in the final analysis it is the performance of the algorithm that is of fundamental importance. Hence we will illustrate many of the algorithms we discuss on real world cybersecurity problems and data and compare and contrast their performance on these data sets, rather than focus more on the validity of the models used in the algorithms. There is a strong bias in this book towards nonparametric methods that relax assumptions about the underlying distribution of the data rather than parametric methods that posit a particular parametric family for the distribution.

As noted above, a single book cannot cover all the topics of mathematics, statistics, artificial intelligence and machine learning that are relevant to cybersecurity. We do not cover game theory, which formalizes the concept of a game and investigates strategies for playing these idealized games. It is possible to model some of cybersecurity as a game with the security analyst "playing" against the attackers. See [310] for an introduction to some of the basic ideas.

We cover only the basics of cryptography; whole books are devoted to this subject. It may be one of the most important aspects of cybersecurity, but it is not a panacea. Ignoring the possibility that quantum computers may make some cryptography obsolete,[5] it is possible to extract information from encrypted traffic, if sufficient care is not taken in the implementation of the cryptographic system. See [491, 492] for some methods of extracting information from encrypted streams through statistical machine learning, and [493] for a potential solution. For another approach to defeating cryptographic protocols without stealing keys or breaking the encryption, the reader should consult, e.g., [273, 274, 284], and the many papers on cryptographic protocol verification.

We also do not discuss the work in artificial immune systems, see for example [106, 194, 422, 432]. This work utilizes analogies with biological systems to develop methodologies to detect "infections" such as malware or malicious users – the *insider threat*.

This is also not a book that covers all of cybersecurity. Chapter 3 provides the basic introduction to cybersecurity and gives an overview of some of the relevant issues. Examples of cybersecurity data and analyses are provided throughout the book to illustrate different methods. For a complete understanding of cybersecurity, a book dedicated to the subject, such as [349, 404], is a place to start.

The best way to learn data analytics is to do it. Several repositories of cybersecurity data are discussed in the text, and the reader is encouraged to obtain any of these data sets and utilize the various methods discussed in the text. The programming languages R [358] and Python have extensive packages that implement the algorithms discussed in the text. Most other languages will also have packages or libraries that implement these methods as well.

[5]Or, they may not. The jury is not yet in on this.

Chapter 2

What Is Data Analytics?

Data analytics usually refers to the analysis of data for business applications, although it is certainly used more generally. Viewed more broadly, it is sometimes referred to as data science, data analysis, or data mining, and in a very real sense one could argue that it is just another name for statistics. However, there are aspects of data analytics that are a superset of statistics, for instance: data storage and retrieval, machine learning, streaming data analysis and frameworks for big data analysis.[1] The term *data science* usually refers to the full spectrum of the data analytic process, including the selection of models, experimental design, and interpretation of the analysis, while *data analytics* refers to the analysis portion.

Leo Breiman [58] split what we would today call data analysts or data scientists into two "cultures": the data modeling culture and the algorithmic modeling culture. David Donoho [123] refers to these as the generative modelers and the predictive modelers. The main distinction of these two cultures is that the data/generative culture is primarily focused on understanding how the data was generated, what models are appropriate to model the data, and how can these models (and parameters fitted to the models) provide useful information about the system that generated the data; the algorithmic/predictive culture is much more interested in the bottom line – how well can my model predict future data? Obviously these are not mutually exclusive, but, like another well-known split – Bayesians versus frequentists – there can be strongly held views on one side or the other. This book leans more toward the algorithmic/predictive end of the scale, although we believe that it is important for users to understand the models underlying their algorithms, and to think carefully about what models (algorithms) are appropriate for a given problem.

Data analytics consists of

1. Ingesting;
2. Cleaning;
3. Visualization and Exploratory Analysis;
4. Feature Extraction and Selection;
5. Modeling;
6. Evaluation;
7. Inference.

[1] One of the authors (DJM) would still argue that this is also mostly statistics, with a smattering of computer science thrown in.

We will consider each of these in turn, although in practice there is a lot of interaction between these. These steps are neither discrete nor independent. For example, there is definitely feedback between modeling and feature extraction, and between both of these and cleaning. How we ingest the data, where and how we store it, and what we collect from the sensor, is driven by the ultimate goal, and also by information learned as we analyze the data. Also, as we will see, visualization and exploratory data analysis (EDA) occurs throughout the process.

The final step, inference, also called exploitation or decision making, is the ultimate goal of the analysis and drives all the previous steps. This is the topic of the rest of the chapters in this book.

It is important to realize that this last step may not be known at the time of collection. In "traditional" statistics[2] one has a scientific or practical question one wants to answer and an experiment is designed to collect and analyze data to answer the question. In cybersecurity, we know the question is something along the lines of "protect the network" – or computer or private data – but from what? One collects very different data to defend against network attacks like denial of service or man-in-the-middle attacks than against phishing, or viruses, or nefarious insiders, or .... What data should we be collecting now, so that we have the data we need to develop protections against the attacks of five years from now? The data volumes and cost of instrumentation to collect data mean that we can't collect everything. Ideally, one always starts with a problem (desired inference) and collects the data to support the inference. In practice, one often relies on existing data, at least initially.

One is often restricted in the data one can collect, either by the vendors that provide the infrastructure of the network, by policy and privacy consideration, or by physical constraints. In reality, we may not even be able to collect the data we would really like to have. For example, if the problem is an insider threat, privacy issues may make it impossible to collect everything the user does. The decisions on what data are collected may be outside the purview of the cybersecurity analyst. The analysis of existing data in the context of the desired inference can be used to argue for changes to allow for the necessary data to be collected, or it can provide information on the fundamental limits on the performance of the inference for the given problem.

This chapter will lay out the basic ideas and provide some illustrations of them for cybersecurity. Each of these ideas will be revisited throughout the book. One aspect of cybersecurity analysis that we would like the reader to keep in mind is that the tools of data analytics provide ways to determine the utility of particular data streams, and can be used to guide decisions of what data should be collected, and how that data should be stored and maintained.

2.1 Data Ingestion

Data comes to the analyst in many ways, not all of them in a nice easily processed form. While in cybersecurity we generally don't have to worry about inputting data scrawled in lab books, we do often have to read system logs (in various formats), extract data from pdfs and web pages, extract information from packet headers and packet payloads (implemented in various protocols and formats), analyze source code and binary files, process unformatted text, analyze network topologies (physical, logical, and defined by various types of communication such as email or social networks); the list of data types and data sources is long and growing.

A very simple example will help set the stage. Many organizations generate reports of cyber intrusions or attacks, and these are often provided as human-readable documents.

[2]There is considerable controversy in the term *traditional statistics*, hence the scare quotes.

These documents do not always follow a strict format, and so writing a parser to extract the data can be challenging. The parser must: convert the pdf to text; locate IP addresses within the document; find and process the section that describes the attack; extract entries from tables, and possibly figures. Even the problem of linking the IP addresses reliably to attacker/victim can be nontrivial.

Ideally, the provider of the data (whether a router, network sniffer, intrusion detection system, or web scraper) has converted the data into a structured and known format, and the problem of ingestion is relatively simple. Even better (possibly) is that the data reside in a database and can be accessed through this database. For online[3] security systems this latter will generally not be available, as the online system will be accessing data directly from a sensor. In this case, it is expected that the sensor will provide the data in a simple and easy to parse format.

For systems that scrape the web, or that monitor email or social network posts or blog posts, ingesting the data can be quite challenging. The data are often in many different formats, with both structured and unstructured components, and in the case of web pages, may be dynamic, with the structure changing in time. In these cases, it may not be simple to determine which data are most relevant.

Relevance depends on the purpose for which the data were collected. In many cases, one collects data which will be used for several different purposes (some of which may not be known at the time of collection), so in addition to the problem of parsing the data is the one of deciding what data to keep and what to discard. For example, in monitoring social networks, does one keep information about the ads that are presented to the user? This will depend on the purpose for which the data are to be used.

2.2 Data Processing and Cleaning

We note at this point that the word “cleaning” carries an unfortunate stigma, implying that the data is “dirty” or in error, and so, as the title of this section indicates, there is more to this step than removing errors. However, some data actually is bad, due to transcription errors or misconfiguration of the sensor, and these need to be removed, or at a minimum identified and labeled as bad. As a rule, we advise against permanently removing “bad” data, although one may temporarily remove it for a given calculation.

If any of the data was input by a human, there is the possibility of error either inadvertent or deliberate. When processing email for spam, phishing, evidence of an insider threat, leaking of proprietary information, one has to cope with the enormous variability in spelling, use of acronyms and short-cuts, slang, typos, dialects, character sets (and emoticons, emojis, and whatever the newest thing is as you read this), the ever growing lexicon, not to mention borrowing from or writing in all the different languages that humans use.

Even data collected automatically can contain errors. Consider the New York Taxi data sets.[4] These contain information about a large number of taxi trips within New York City and the environs, with pickup/dropoff times, latitudes and longitudes for the pickups and dropoffs, and other data. Many of the latitudes and longitudes are reported as $(0, 0)$ which is off the coast of Africa – some GPS units report this when they malfunction or are unable to obtain a GPS signal. This is not the only error in the GPS data, however – there are values in Europe and other U.S. states – and even for those values which are theoretically possible, such as a position in Maine, the reported cost proves that these are indeed errors.

[3]An online, or streaming, security system is one which collects data as it arrives (packets over the network, emails as they are delivered, keystrokes as they are typed) and processes each observation for an immediate decision (real-time or near real-time). Typically, online systems do not store and retrieve older data, although some short-term memory is often available.

[4]Available at http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml.

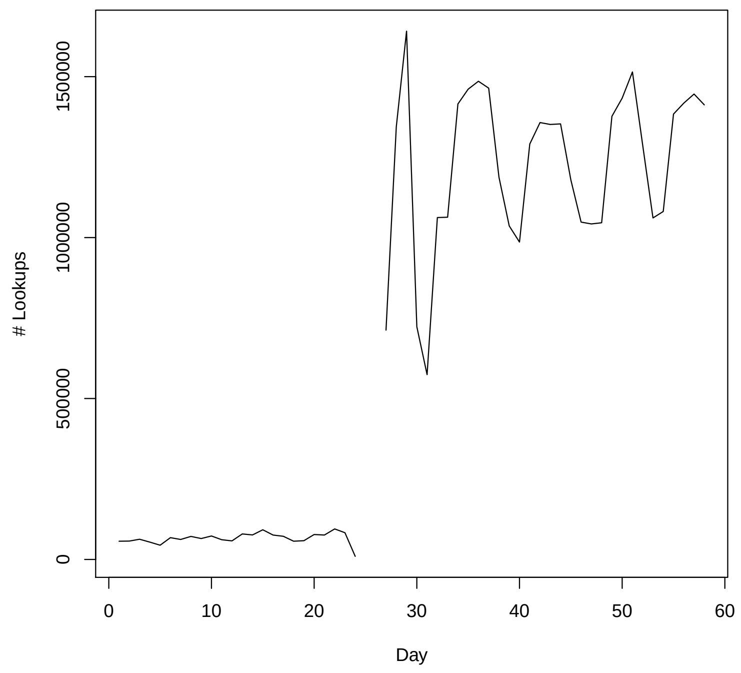


Figure 2.1: Number of DNS lookups by day.

Some of the pickup times occur after the dropoff times, which is likely to be a result of human error, but may also be an error in the device that records the times.

These data also have the problem that different companies provide the data in slightly different formats or with slightly different values. Even with a well-curated data set, there can be a number of problems that must be addressed before any inferences can be made.

It is also not always evident which parts of the data are errors or need to be “cleaned”. Consider the Domain Name Service (DNS) data[5] [232, 233]. The data correspond to 40, 821, 591 DNS lookups between 15, 296 computers over a period of 58 days. The first three lines of the data file (“dns.txt”) are:

```
2,C4653,C5030
2,C5782,C16712
6,C1191,C419
```

The first number represents time, the second string is the source computer and the third is the label of the computer resolved. Figure 2.1 gives an overview of the data.

Several things are clear from the picture. First, there is a change that occurs after day 24 – there are two missing days, and then the volume of data increases dramatically. Second, in that later half of the data, one can clearly see a periodicity of 7 days – a weekly pattern.

These data are very clean. They have been curated by the researchers at Los Alamos and contain no errors (that are detectable by us). However, the data may still need to be cleaned, depending on the purpose for which they are to be used.

[5]Available at http://csr.lanl.gov/data/cyber1/.

One can view the DNS lookups as evidence of communications between the source and destination systems. Consider the problem of determining which computer is the most active. First one needs to define "active". If one defines it as the total number of DNS lookups, the most active computer is C585 with $1,624,999$ requests. The next two are C743 with $1,149,807$ requests and C1823 with $391,719$ requests. Note that these data contain times where a source looks up itself. This happens $865,039$ times in these data! It might be reasonable for some purposes to remove these – "clean" them – even though they are not errors. In this example, this kind of cleaning does not make a difference in the results.

Another possible definition of "active" is the computer that connects (looks up) the most distinct computers. There are $425,025$ unique records, so on average a computer looks up another specific computer about 96 times. Under this definition, the most active computer is C561 connecting to $12,103$ other computers with the next two most active computers being C395 connecting to $11,920$ computers, and C754 connecting to $1,067$ computers.

This example illustrates two of the most common preprocessing or cleaning methods: removing duplicates and removing observations not relevant or useful for the inference task, either because they contain errors, or in this instance because they correspond to something we choose not to consider.[6]

Another interesting data set available from Los Alamos is the network flows data.[7] These are connections between computers containing the time and duration of the connection, the source and destination machines and ports, the protocol used, and the number of packets and bytes. Source and destination ports are integers between 0 and 2^{16}; however, a perusal of the data shows that many of these start with the letter "N." This is a flag indicating that the port has been anonymized to protect information specific to this network. This points out a problem that cybersecurity analytics must grapple with: it is difficult to provide data that is timely, broadly useful, and which does not provide information specific to the system that can be used to compromise it. There are $129,977,412$ flows in these data, each flow corresponding to one direction of a flow of data between two computers.

2.3 Visualization and Exploratory Analysis

Exploratory data analysis (EDA) is the most important aspect to any data analysis. One cannot properly model data without understanding the data, and we are a visual species that obtains most of our information through our eyes. While one can use many machine learning tools as black box algorithms – blindly apply them to a data set and see what comes out – this is generally not a good idea. By the same token, one can cross streets without obeying the traffic laws or looking for oncoming traffic. We strongly advise against either of these decisions.

2.3.1 Scatterplots

The simplest, most often used, and in many ways most powerful EDA tool is the simple scatterplot. Here we plot as dots all the points of two-dimensional data. This idea can be extended to plots of multivariate data, although the more variables one plots, the more difficult the interpretation.

Consider the network flow data from the previous section. Two of the fields are the number of bytes and the number of packets in the session. Logically, these should be

[6]One could view this latter as more of a subsetting problem. This is true, but it is an example of a preprocessing step. For another example: in the taxi data, one might remove data whose pickup and dropoff locations are the same (within some tolerance), or whose locations fall outside a given distance from the center of New York City.

[7]Also available at http://csr.lanl.gov/data/cyber1/.

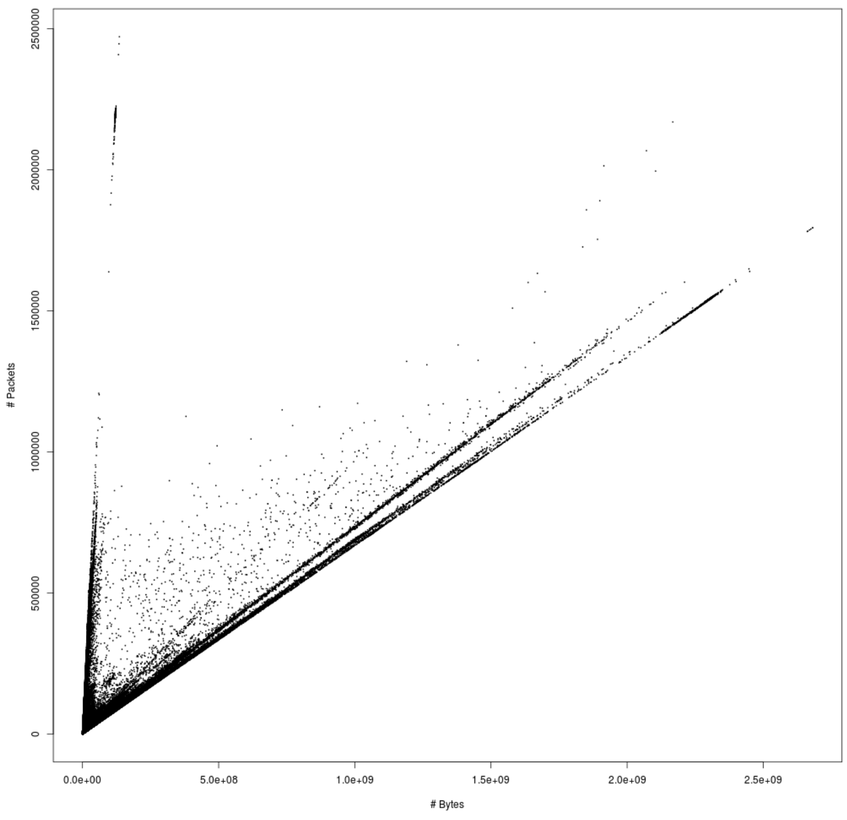
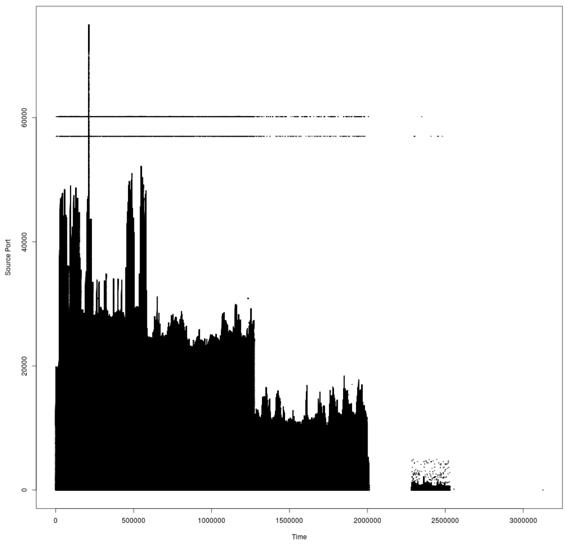


Figure 2.2: Number of bytes versus the number of packets in the flows data.

(approximately) linearly correlated, and we can investigate this hypothesis through a scatterplot. Figure 2.2 plots the number of bytes against the number of packets. We observe distinct lines, with slope corresponding to a relatively constant number of bytes per packet. A hypothesis is that the different lines correspond to different applications.

Next consider the ports used by the flows. Ports can be considered categorical (certain ports are assigned to certain applications and vice versa) and ordinal (port ranges are used freely as source or destination ports for any service that needs a port). Figure 2.3 shows the source and destination ports in time. Three things are immediately obvious. First, these plots look identical (to the eye) – this is because these data correspond to two-way communications, and a source port for one way is a destination port going the other way. This shows up clearly in these scatterplots. Second, there are definite differences in time in the pattern of the ports in each of these plots, as seen by the fact that the gap between the largest observed values of the port numbers and the majority of the observed values increases in time. Finally, the full range of ports are not used. Note that we are not claiming that this is an error in the data – this type of analysis can only provide information about the data; any explanations must come from the data source.

The scatterplot tells us quite a bit about the data. As discussed above, in [233] we are informed that while some ports (ones commonly tied to a set of specific common applications) are reported accurately, many of the ports are "anonymized." This anonymization process is not described, but it is not necessarily the case that the anonymization is the



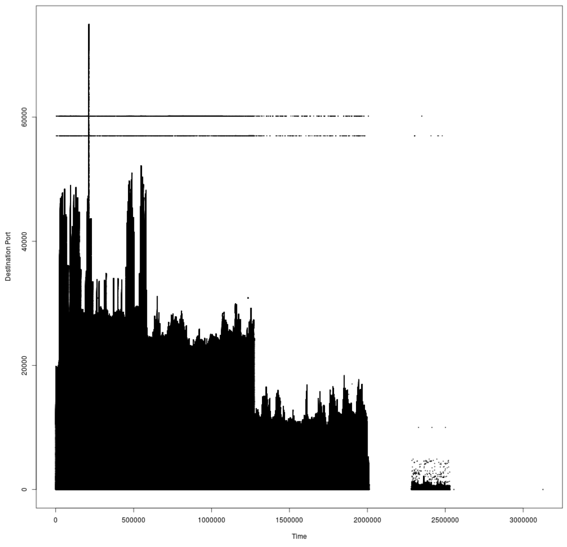


Figure 2.3: Source ports and destination ports in time.

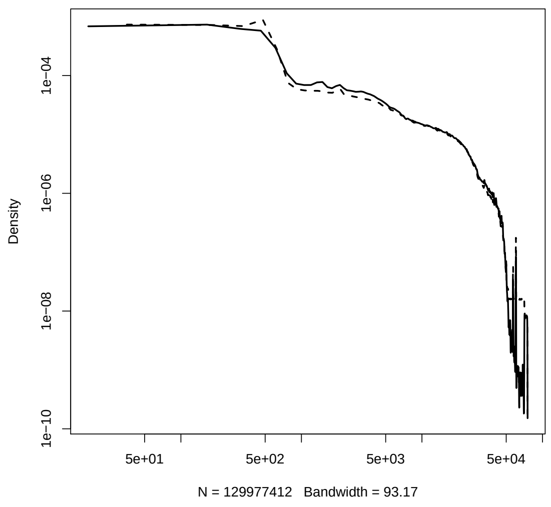


Figure 2.4: Estimate of the probability density function of ports (source ports in the solid curve, destination ports in the dashed curve) on a log-log scale for the network flow data.

reason for the changes in time we observe in Figure 2.3. Whether the differences are the result of changes in the anonymization process or in the flows on the network cannot be known just from this analysis.

In Figure 2.4 we depict the probability density functions (estimated using a kernel estimator, see Chapter 4) for the source and destination ports. There is a dip around 500 and another around $10,000$. The ports less than 500 are pretty much all assigned, and thus likely to be left alone by the anonymization process, and also likely to be commonly seen on the network, as is indicated by the flatness of the curves in this region.

2.4 Pattern Recognition

Before considering the rest of the topics in this chapter, we need to introduce the key set of ideas that allow us to design algorithms to detect cyber attacks. These ideas will be revisited in more detail in Chapters 5 and 6.

We are interested in malware detection and classification, phishing, detecting an insider threat, detecting unusual activity, and characterization of activities and network sessions. The types of pattern recognition that we are typically interested in for cybersecurity fall into two general categories: Classification (also called supervised learning) and clustering (a form of unsupervised learning). We are also concerned with outlier detection and with the characterization of "normal" activities, which properly fall into the more general area of statistics.

In classification we have an input (the data) and an output (the class label). There is a version of supervised learning that is distinct from classification: regression. Here the output is a value (which may be continuous or discrete) that we want to match to the input. For example, the reader is probably familiar with linear regression, where the goal is to fit a linear model that predicts the output from the input. For example, consider predicting the number of packets used to transfer a given number of bytes in a flow. See Figure 2.6. Obviously there is more than one line here, and so one must subset the data – by the direction of the flow, for example. The figure indicates that, with a little work, it may well be possible to design a set of algorithms that perform this prediction, which may be useful in network design.

2.4.1 Classification

A classification algorithm is one which takes data (features extracted from some set of data) and assigns one of a set of class labels to each observation. This can be a "hard" assignment – the algorithm returns a single class label – or a "soft" assignment – the algorithm returns a set of class labels with weights corresponding to the algorithm's assessment of the probability that the observation was generated from the given class.

Classification starts with training data. These are observations for which the true class is known. Given these labeled observations, the goal is to produce an algorithm that, to the extent possible, will label new (unlabeled) observations correctly.

Formally, we are given a training set $(x_1, y_1), \ldots, (x_n, y_n)$. The $x_i \in \mathcal{X}$ are observations (features), the $y_i \in \mathcal{Y}$ are class labels. We seek to find a function $g : \mathcal{X} \to \mathcal{Y}$ that is "good". We'll consider some definitions of "good", and how to estimate them, in Section 2.7. For now, we might want to minimize the probability of incorrect classification (the error):

$$P[g(X) \neq Y]. \tag{2.1}$$

To do this, we'd like to solve

$$\arg\max_g P[g(X) \neq Y]. \tag{2.2}$$

That is, we want g to be the *Bayes classifier*: the classifier with the lowest error, the best probability of correct classification.

We may at times consider different criteria for performance. For example, the cost of incorrectly flagging an email as phishing may be very different from missing the phishing attempt, and so we may assign different weights to the different types of errors, or we may allow more errors of one type in order to reduce the number of errors of the other type. We will discuss this in more detail below.

2.4.2 Clustering

Clustering, or unsupervised learning, takes unlabeled data and returns a grouping of the data. We are not given any a priori grouping or class labels. Instead we seek to find the "natural" groups, called *clusters*, within the data.

As might be suspected, this is not well defined. First, what does it mean to be a cluster? Consider Figure 2.5. How many clusters are in this figure, and what is a cluster? Clearly there are "round-ish" clusters (4 of them?). There are also "chains", but it is not clear how many of these there are – is the central "plus" a single cluster, or 4 (or more) "chain" clusters? Are the arms of the spiral 3 clusters, or more?

Now consider Figure 2.5 at a higher level: clearly there are three "groups" in this: the plus, the spiral, and the balls. This illustrates the hierarchical nature of clustering – there may be different clusterings at different scales, and thus a hierarchy of clusters.

Clustering thus has two fundamental, and difficult, problems. First, one must define what one means by a cluster, and the definition will drive what types of clusters that one can find. Then, one has to figure out how many clusters there are. Neither of these tasks is trivial, and while there are many tools that aid in this analysis, there is no universally optimal procedure. This is in contrast to classification, in which there are universal classifiers – those which attain the optimal classification performance – with the nontrivial caveat that this is an asymptotic statement, which may not be particularly useful in the real world of finite training sets.

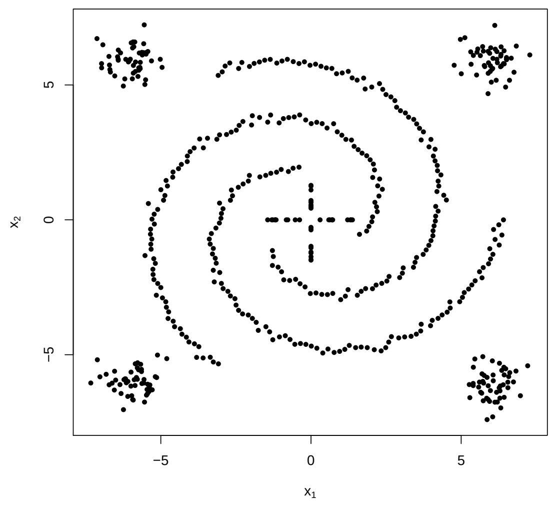


Figure 2.5: Random data illustrating one of the difficulties in defining “clusters” in data.

2.5 Feature Extraction

A measurement is a value extracted from some object that one is investigating. For example, in health care this might be blood pressure or temperature or some aspect of health history. A feature is a function (possibly the identity function) of one or more measurements that is to be used to perform inference. Given a problem domain – such as network traffic – there are potentially many measurements one could take, some more useful for a given inference task than others. Given a set of measurements, there are many (infinitely many) features one could produce. Feature extraction will refer to both choosing the appropriate measurements to extract, and determining the features that are relevant to the problem. Feature selection is choosing the best – under some criterion – of features to use for inference.

Consider the problem of monitoring network traffic to detect attacks on the network. The data we observe, at the most basic level, are packets. There are values that we need to extract from these packets – port numbers, IP addresses, protocol flags, byte counts, packet payload information. Further, we may be interested in aggregates of packets, such as TCP/IP sessions or flows. Each measurement we choose to take is a feature. Further, we can combine features, such as taking the number of bytes and dividing by the number of packets in a flow to produce a new feature – the bytes-per-packet ratio.

Feature extraction is very much driven by expert knowledge of both the problem domain and the specific problem. For example, if one is interested in detecting distributed denial of service attacks, features such as the number of connection attempts that are not completed might be a useful feature, but the number of connections containing source ports above $30,000$ may not be particularly useful.

2.5.1 Feature Selection

Even when the data analyst is herself a domain expert, or has access to one, there will be features that the expert knows are important, and others that are unnecessary for the desired inference, or that are redundant. Also, each feature we measure contains some amount of noise, either a purely random process – such as the time between packets or the actual value of a source port assigned randomly – or best modeled as random – such as the number of packets in a particular web-flow, or the duration of an SSH connection.

It may seem obvious that the more features we collect, the better we can do at inference. This is false, and not just because we are imperfect humans who cannot utilize the features optimally. Trunk [211, 436] showed that even if a new feature that we might wish to collect has valuable information for the inference task that no other variable contains, and we use the optimal model for the inference, it can be the case that utilizing this new variable will reduce the quality of the inference. In other words, sometimes adding new useful information can hurt more than it helps.

This seemingly counter-intuitive statement is an example of the bias-variance trade-off, which we will discuss in more detail in Chapter 4. The intuition comes from the fact mentioned above that features contain noise. The idea is that if the amount of noise is high compared to the benefit of the feature, this noise can add enough "confusion" to our estimates to overcome any benefit of the new feature. Thus, when we add features, we want them to do more than add useful information, we need them to add enough useful information to overcome the inherent variability (noise or error) in the feature.

There are a number of feature selection methods, and they fall into two main types: true selection methods seek to choose a subset of the features among all the features measured, and projection or embedding methods compute linear (or nonlinear) combinations of the features, and then select a small set of these combinations.

It should be noted that feature selection is never performed in a vacuum. It is important to keep the ultimate goal in mind: what inference are you trying to make? Features that are ideal for detecting phishing in email are likely to be useless for detecting an insider emailing proprietary information from the company.

Suppose one observes a session between two computers and wishes to determine whether the session is web traffic. The obvious thing to do is look at the ports – ports 80, 8080 and 443 are the most common web ports. In this example, however, we observe traffic between unknown ports, and suspect a covert web server has been set up. Assume we do not observe the packet payload – all we observe are the time of the session, the duration of the session, number of packets and number of bytes. We'll use the flows data to investigate this, as an illustration of feature selection methods.

The time of the sessions is in seconds from an arbitrary and unknown start time, so for the purposes of this study, we will assume that the first packet in the data occurred at time $0:00$ and convert all the times to hour of the day from this baseline, taking on integer values from 0 to 23. The method we will use for determining what type of session we are observing is to compute the mean of each of the features for web traffic – those in our data to/from web ports – and nonweb traffic – everything else. To do this properly, we would use a subset of each of these data sets to construct the estimates, then use the other subsets to compute our performance under different choices of features. Since we are merely illustrating the ideas, we won't bother to do this. Our estimates are a type of "resubstitution" estimate, which we'll describe below, and give a biased estimate of performance, but because of the large numbers of observations this bias is not of large concern for this example.

To be precise, we will compute the following statistic:

$$c = \arg\min_{m} d(x, m), \tag{2.3}$$

where the minimum is taken over the two means, web and nonweb traffic, and d is the euclidean distance. So we compute the distances to the means shown in Table 2.1, and whichever mean is closest to our data determines whether we call the traffic web or not.