~~digits as well as the ‘truth’ values ‘True’ and ‘False’ in classical logic.~~ These numbers are readable as symbolic logical propositions governed by the rule $Y = \neg X_1 \wedge X_2 \wedge X_3$. The table acts as a hinge between numbers and symbolic thought or cognition. Third, the NAND table in particular has an obvious operational relevance in digital logic, since digital circuits of all kinds – memory, processing, and communication – comprise such logical functions knitted together in the intricate gated labyrinths of contemporary calculation.[1]

A pre-machine learning programmer, tasked with implementing the logical NAND function might write:

Listing 1.1: XOR function in Python

```
Y = not(X1 & X2 & X3)
```

The trivial simplicity of the code stands out. This looks like the kind of symbol manipulation that that computers can easily be programmed to do. How, by contrast, could such a truth table be learned by a machine, even a machine whose *modus operandi* and indeed whose very fabric is nothing other than a massive mosaic of `NAND` operations inscribed in semiconductors? Machine learning of on elementary truth tables is not a typical operation today, but usefully illustrates something of the diagrammatic transformations that programming or coding (and classical AI) has undergone .

Machine learners such as decision trees or neural nets typically know nothing of the logical calculus and its ~~elementary logical operations~~. Can they be induced to learn ~~it~~? A perceptron, an elementary machine learner dating from the 1950s, that ‘learns’ the binary logical operation NAND (Not-AND)

[1] The philosophy Charles Sanders Peirce himself had first shown that combinations of NAND operations could stand in for any logical expression whatsoever, thus paving the way for the diagrammatic weave of contemporary digital memory and computation in all their permutations. Today, NAND-based logic is norm in digital electronics. NOR – NOT OR – logic is also used in certain applications.

is expressed in twenty lines of python code on the Wikipedia 'Perceptron' page (*Perceptron* 2013) (see listing ~~1.2~~). It is a standard machine learner almost always included in machine learning textbooks and usually taught in introductory machine learning classes. ~~The code outputs a series of numbers shown in table 1.2.~~

Listing ~~1.2~~: Perceptron learning *XOR* function

```
threshold = 0.5
learning_rate = 0.1
weights = [0, 0, 0]
training_set = [((1, 0, 0), 1), ((1, 0, 1), 1), ((1, 1, 0), 1),
    ↪ ((1, 1, 1), 0)]

def dot_product(values):
    return sum(value * weight for value, weight in zip(values,
        ↪ weights))

out = 'weight1,␣weight2,␣weight3,␣error_count,␣iteration'
print out
iteration_count = 0
while True:
    error_count = 0
    iteration_count += 1
    for input_vector, desired_output in training_set:
        out = ','.join(map(str,weights)) + ',' + str(error_count)
        out = out + ',' + str(iteration_count)
        result = dot_product(input_vector) > threshold
        error = desired_output - result
        print out
        if error != 0:
            error_count += 1
            for index, value in enumerate(input_vector):
                weights[index] += learning_rate * error * value
    if error_count == 0:
        break
```

weight1	weight2	weight3	error_count	iteration
0.00	0.00	0.00	0	1
0.10	0.00	0.00	1	1
0.20	0.00	0.10	2	1
0.30	0.10	0.10	3	1
0.30	0.10	0.10	0	2
0.40	0.10	0.10	1	2
0.50	0.10	0.20	2	2
0.50	0.10	0.20	2	2
0.40	0.00	0.10	0	3
0.50	0.00	0.10	1	3
0.50	0.00	0.10	1	3
0.60	0.10	0.10	2	3
0.50	0.00	0.00	0	4
0.60	0.00	0.00	1	4
0.60	0.00	0.00	1	4

Table ~~1.2~~: Iterative change in weights as a perceptron learns the NAND function

What does the code in listing ~~1.2~~ show or say about the transformation in programmability or the writing of programs? First of all, we should note the relative conciseness of the code vignette. Much of the code here is familiar, generic programming. It defines variables, sets up data structures (lists of numerical values), checks conditions, loops through statements or prints results. In citing this code, I am not resorting to a technical publication or scientific literature as such, or even to a machine learning software library or package (in `scikit-learn`, the same model could be written `p = scikit-learn.linearmode.Perceptron(X,Y)`), just to a Wikipedia page, and the relatively generic and widely used programming language `Python`.[^1.31] Whatever the levels of abstraction associated with machine learning, the code is hardly ever hermetically opaque. As statements, everything lies on the surface.

Second, while the shaping of data, the counting of errors, and the optimisation of models are topics of later discussion, the code shows some typical features of a machine learner in the form of elements such the `learning rate`, a `training_set, weights`, an `error count`, and a loop function that multiplies values (`dot_product`). Some of the names such as `learning_rate` or `error_count` present in the code bear the marks of the theory of learning machines that we will discuss.

Third, executing this code (by copying and pasting it into a `Python` terminal, for instance) produces several dozen lines of numbers. They are initially different to each other, but gradually converge on the same values (see table ~~1.2~~). These numbers are the 'weights' of the nodes of the perceptron as it iteratively 'learns' to recognise patterns in the input values. None of the workings of the perceptron need concern us at the moment. Again, what runs across all of these observations are the numbers that the algorithm produces as output – they embody the program the perceptron has written. How has the learning happened in code? The NAND truth table has been re-drawn as a dataset (see line 4 of the code that defines the variable `training_set`). The perceptron has learned the data by approaching it as a set of training examples, and then adjusting its internal model – the weights that are printed during each loop of the model as the output – repeatedly until the model is producing the correct result values Y of the truth table. The algorithm exits its main loop (`while True:`) when there are no errors.

The perceptron algorithm computes numbers - `0.79999`, `2.0` – as weights or parameters. These numbers display no direct correspondence with the symbolic categories of boolean True and False or the binary digits `1` and `0`. There may be a relation but it is not obvious at first glance. The problem of mapping these calculated parameters – and they truly abound in machine learning – triggers many different diagrammatic movements in machine learning (and these different movements will be discussed in chapters ~~**??**~~ and ~~**??**~~). These numbers engender much statistical ratiocination (see chapter ~~**??**~~). Here we need only note the contrast between symbolically organised statements like the NAND truth table of Table ~~1.1~~ and the operational statements in table ~~1.2~~.

The operation here is recursive: a model or algorithm implemented in digital logic (as `Python` code) has ‘learned’ – a term we need to explore more carefully – a basic rule of digital logic (the `NAND` function) ~~at least approximately~~ by treating logical propositions as data. This mode of transformation is symptomatic. The learning done in machine learning has few cognitive or symbolic underpinnings. It differs from classical AI in that it treats existing symbolic, control, communicative and increasingly, signifying processes (such as the cat faces that `kittydar` tries to find), and latches onto them programmatically only in the form of weights.

The elements of machine learning

If this growing of programs through modelling data is a different mode of coding operations, and a different mode of knowing, how does one learn to do it? In the course of writing this book, as well as reading the academic textbooks, popular how-to books, software manuals, help documents and blog-how-to posts, I attended graduate courses in Bayesian statistics, genomic data analysis, data mining, and missing data. I also participated in online machine learning courses and some machine learning competitions. These are all widely shared activities for people learning to do machine learning. I watched and copied down by hand many statements, equations and propositions from Youtube videos of the eighteen lectures in Andrew Ng’s Stanford University lectures CS229 computer science course recorded in 2008. These lectures have cumulative viewing figures of around 500,000 (*Lecture 1 | Machine Learning (Stanford)* 2008). I spent extended hours learning to use various code libraries, packages and platforms in `R`, `Python` and to a limited extent `Torch`~~.~~ Finally, I gradually accumulated and worked with a set of around 400,000 articles drawn from various fields of science in response to search queries on particular machine learners such as `support vector machine` or `expectation maximization`. This

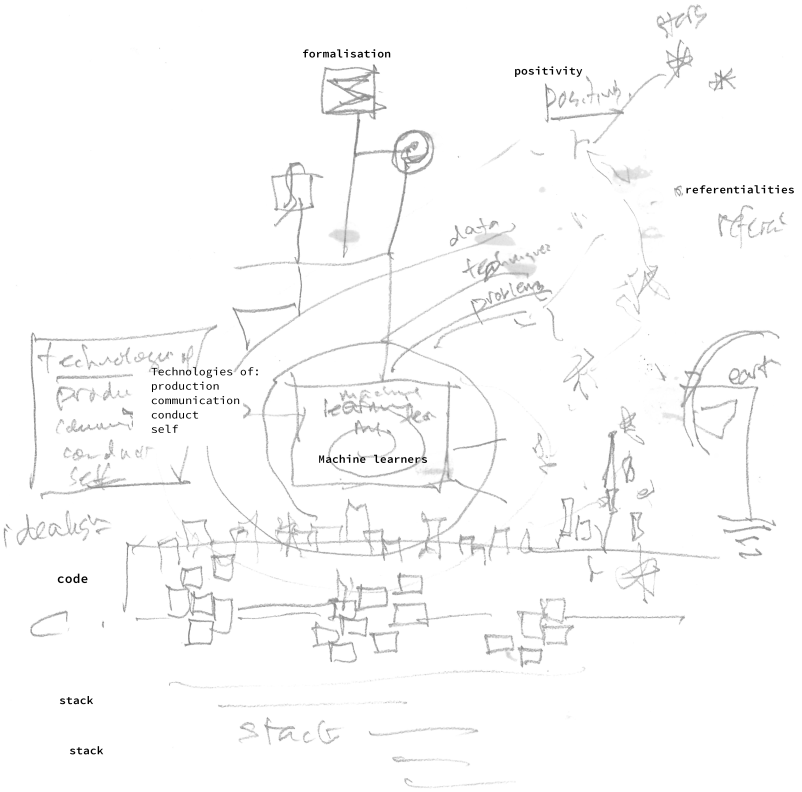


Figure 1.1: Elements of learning to machine learn

accumulation of knowledge appears in ~~figures 1.1~~ as a network-cloud.

Some broadly shared topic structures help in ~~any navigation of~~ the software libraries, pedagogical and research literatures. The textbooks, the how-to recipe books (Segaran 2007; Kirk 2014; Russell 2011; Conway and White 2012) and the online university courses on machine learning often have a similar topic structure. They nearly always begin with 'linear models' (fitting a line to the data), then move to logistic regression (a way of using the linear model to classify binary outcomes; for example, spam/not-spam; malignant/benign; cat/non-cat), and afterwards move to some selection of neural networks, decision trees, support vector machine and clustering algorithms. They add in some decision theory, techniques of optimization, and ways of selecting predictive models (especially the bias-variance trade-off).[2] The topic structures have in recent years started to become increasingly uniform. This coagulation around certain topics, problems and mathematical formalisms is both something worth analyzing (since, for instance, it definitely affects how machine learning is taken up in different settings), but should not be taken as obvious or given since it results from many iterations.

Amidst this avalanching machine learning materials and practice, a single highly cited and compendious textbook, *Elements of Statistical Learning: Data Mining, Inference, and Prediction* dating from from around 2000 and

[2] They differ, however, in several important respects. Reading the *Elements of Statistical Learning* textbook or one of machine learning books written for programmers (for example, *Programming Collective Intelligence* or *Machine Learning for Hackers* (Segaran 2007; Conway and White 2012)) does not directly subject the reader to machine learning By contrast, doing a Coursera course on machine learning brings with it an ineluctable sense of being machine-learned, of oneself becoming an object of machine learning. The students on Coursera are the target of machine learning. Daphne Koller and Andrew Ng are leading researchers in the field of machine learning, but they also co-founded the online learning site Coursera. As experts in machine learning, it is hard to imagine how they would not treat teaching as a learning problem. And indeed, Daphne Koller sees things this way:

> There are some tremendous opportunities to be had from this kind of framework. The first is that it has the potential of giving us a completely unprecedented look into understanding human learning. Because the data that we can collect here is unique. You can collect every click, every homework submission, every forum post from tens of thousands of students. So you can turn the study of human learning from the hypothesis-driven mode to the data-driven mode, a transformation that, for example, has revolutionized biology. You can use these data to understand fundamental questions like, what are good learning strategies that are effective versus ones that are not? And in the context of particular courses, you can ask questions like, what are some of the misconceptions that are more common and how do we help students fix them? (*What We're Learning from Online Education | Video on TED.com* 2012)

Whether the turn from 'hypothesis-driven mode to the data-driven mode' has 'revolutionized biology' is debatable (I return to this in a later chapter). And whether or not the data generated by my participation in Coursera's courses on machine learning generates data supports understanding of fundamental questions about learning also seems an open question. Nevertheless, the loopiness of this description interests and appeals to me. I learn about machine learning, a way for computer models to optimise their predictions on the basis of 'experience'/data, but at the same time, my learning is learned by machine learners. This is not something that could happen very easily with a printed text, although versions of it happen all the time as teachers work with students on reading texts. While Coursera and other MOOCs promise something that mass education struggles to offer (individually profiled educational services), it also negatively highlights the possibility that machine learning in practice can, somewhat recursively, help us make sense of machine learning as it develops.

currently in its second edition (Hastie, Tibshirani, and Friedman 2009), can be seen from almost any point of the terrain.[3] At least for archaeological purposes, I regard this book as an assemblage, and a diagram that presents many important statements, forms of visibility and relations between forces at work in machine learning. The authors of the book, Jeff Hastie, Rob Tibshirani and Jerome Friedman are statisticians working at Stanford and Columbia University.

Elements of Statistical Learning is a massive textual object, densely radiant with equations, tables, algorithms, graphs and references to other scientific literature. From the first pages proper of the book, almost every page has a figure or a table or a formal algorithm (counting these together: 1670 equations; 291 figures; 34 tables; and 94 algorithms, giving a total of 2089 operational statements threaded through the book). Equations rivet the text with mathematical abstractions of varying sophistication. On each page of the book we are seeing, reading, puzzling over and perhaps learning from the products of code execution. The graphic figures are all produced by code. The tables are mostly produced by code. The algorithms specify how to implement code, and the equations diagram various operations, spaces and movements that meant to run as code.

In the range of references, combinations of code, diagram, equation, scientific disciplines and computational elements, and perhaps in the somewhat viscous, inter-objectively diverse referentiality that impinges on any reading of it, *Elements of Statistical Learning* betrays some hyperobject-like positivity (Morton 2013). It is an accumulation of forms, techniques, practices, propositions and referential relations. *Elements of Statistical Learning*

[3] The complete text of the book can be downloaded from the website http://statweb.stanford.edu/~tibs/ElemStatLearn/. At the end of short intensive course on data mining at the Centre of Postgraduate Statistics, Lancaster University, the course convenor, Brian Francis, recommended this book as the authoritative text. Some part of me rues that day. That book is a poisoned chalice; that is, something shiny, valuable but also somewhat toxic.

combines statistical science with various algorithms to ‘learn from data’ (Hastie, Tibshirani, and Friedman 2009, 1). The data range across various kinds of problems (identifying spam email, predicting risk of heart disease, recognising handwritten digits, etc.). The learning takes the form of various machine learning techniques, methods and algorithms (linear regression, k-nearest neighbours, neural networks, support vector machines, the Google Page Rank algorithm, etc.).

There are other juggernaut machine learning textbooks. Ethem Alpaydin’s *Introduction to Machine Learning* (Alpaydin 2010) (a more computer science-base account), Christopher Bishop’s heavily mathematical *Pattern ~~r~~ecognition and ~~m~~achine ~~l~~earning* (Bishop 2006), Brian Ripley’s luminously illustrated and almost coffee-table formatted *Pattern Recognition and Neural Networks* (Ripley 1996) , Tom Mitchell’s earlier artificial intelligence-centred *Machine learning* (Mitchell 1997)~~,~~ Peter Flach’s perspicuous *Machine Learning: The Art and Science of Algorithms that Make Sense of Data* (Flach 2012) , or further afield, the sobering and laconic *Statistical Learning for Biomedical Data* (Malley, Malley, and Pajevic 2011) all cover a similar range of data and approaches. These and quite a few other recent machine learning textbooks display a range of emphases, ranging from the highly theoretical to the very practical, from an orientation to statistical inference to an emphasis on computational processes, from science to commercial applications.

Who reads machine learning textbooks?

How ~~does such textbook~~ help us assess and engage with claim to learn from data or to produce knowledge differently? While ~~it~~ certainly does not comprehend everything taking place in and around machine learning, it diagrams several *elementary* tendencies or traits. It's readership as we will see is widespread. It has a heterogeneous texture in terms of the examples, formalisms, disciplines and domains it covers. It starkly renders the problems of making sense of mathematical operations, diagrams and transformations carried on through calculation, simulation, deduction or analysis. It draws on a matrix of operational practices, particularly in the form of the `R` code it heavily but somewhat latently relies on. In short, *Elements of Statistical Learning* presents a multi-faceted and somewhat monumental ~~layering~~ of abstractive practice that might be open to archaeological inquiry.

Citations	Field
556	Engineering
520	Computer Science
380	Engineering", "Computer Science
369	Biotechnology & Applied Microbiology
344	Mathematical & Computational Biology
207	Mathematics", "Computer Science
175	Chemistry
150	Remote Sensing
116	Instruments & Instrumentation
99	Imaging Science & Photographic Technology", "Computer Science
81	Engineering", "Automation & Control Systems
77	Mathematics", "Biochemistry & Molecular Biology
77	Public, Environmental & Occupational Health
75	Medical Informatics
69	Mathematics
69	Operations Research & Management Science", "Computer Science
67	Research & Experimental Medicine
66	Geology
62	Marine & Freshwater Biology
60	Engineering", "Engineering

Table ~~1~~.3: Subject categories of research publications citing *Elements of Statistical Learning* 2001-2015. The subject categories derive from Thomson-Reuter *Web of Science.*

Who reads the *Elements of Statistical Learning*? It is often cited by academic machine learning practitioners as an authoritative guide. On the other hand, students participating in new data science courses often come from different disciplinary backgrounds and find the tome unhelpful (see the comment by students during an introductory data science course documented in (Schutt and O'Neil 2013)). Whether the citations are friendly or not, it is hard to find a field of contemporary science, engineering, natural, applied, health and indeed social science that has not cited it. A Thomson-Reuters Scientific 'Web of Science'(TM) search for references citing either the first or second edition of (Hastie, Tibshirani, and Friedman 2009) yields around 9000 re-

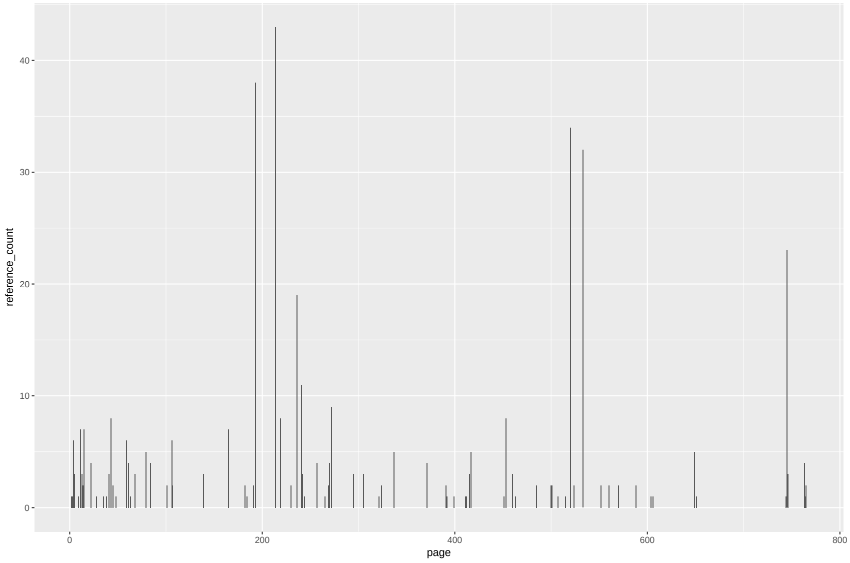


Figure ~~1~~.2: Pages cited from *Elements of Statistical Learning* by academic publications in all fields.

sults. These publications sprawl across over 100 different fields of research. While computer science, mathematics and statistics dominate, a very diverse set of references comes from disciplines from archaeology, through fisheries and forestry, genetics, robotics, telecommunications and toxicology ripple out from this book since 2001. Table ~~1.3~~ shows the top 20 fields by count. One could learn something about the diagrammatic movement of machine learners from that reference list, which itself spans biomedical, engineering, telecommunications, ecology, operations research and many other fields. While it is not surprising to see computer science, mathematics and engineering appearing at highest concentration in the literature, molecular biology, control and automation, operation research, business and public health soon appear, suggesting something of the propagating accumulation or positivity of machine learning.

So we know that *Elements of Statistical Learning* passes into many fields. But what do people read in the textual environment of book? In general, the thousands of citations of the book themselves compose a diagram of the book's intersection with different domains of ~~knowledge~~. The relative concentrations and sparsities of these citations suggest there may be specific sites of engagement in the techniques, approaches and machines that the book documents. Of the around 760 pages in the two editions ((Hastie, Tibshirani, and Friedman 2009) and (Hastie, Tibshirani, and Friedman 2001)), around 83 distinct pages are referenced in the citing literature. As figure ~~1~~ 2 indicates, certain portions of the book are much more heavily cited than others. This distribution of page references in the literature that cites *Elements of Statistical Learning* is a rough guide to how the book has ~~been read in~~ different settings. For instance, the most commonly cited page in the book is page 553. That page begins a section called 'Non-negative Matrix Factorization', a technique frequently used to process digital images to compress their visual complexity into a simpler set of visual signals (Hastie, Tibshirani, and Friedman 2009, 553). (The underlying reference here is the highly cited paper (Lee and Seung 1999)) Like `kittydar`, it, as Hastie and co-authors write, 'learns to represent faces with a set of basis images resembling parts of faces' (Hastie, Tibshirani, and Friedman 2009, 555). (So `kittydar`, which doesn't use NMF, might do better if it did, because it could work just with parts of the images that lie somewhere near the parts of a cat's face – its nose, its eyes, its ears.)

Conversely, what do the authors of *Elements of Statistical Learning* read? The book gathers elements from many different quarters and seeks to integrate them in terms of statistical theory. The hyperobject-like aspect of the book comes from the thick weave of equations, diagrams, tables, algorithms, bibliographic apparatus, and numbers wreathed in typographic ornaments drawn from many other sources. For instance, in terms of outgoing references or the literature that it cites, *Elements of Statistical Learning* webs together a field of scientific and technical work with data and

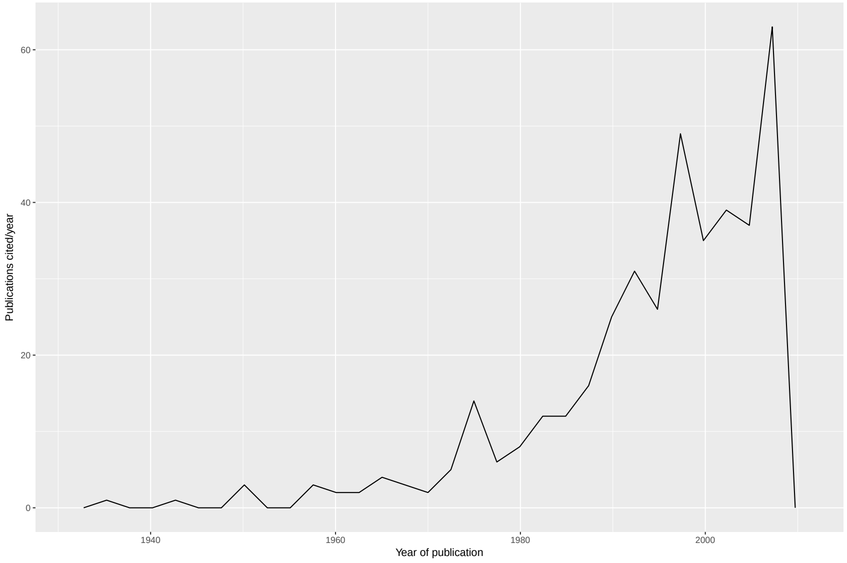


Figure 1.3: Publications cited in*Elements of Statistical Learning*. The references range over almost 80 years, with peaks in late 1970s, late 1980s, mid-1990s and mid-2000s. These peaks relate to different mixtures of cybernetics, statistics, computer science, medicine, biology and other fields running through machine learning. Regression-related publications form the main body of citation.

predictive models ranging across half a century. The reference list beginning at page 699 (Hastie, Tibshirani, and Friedman 2009, 699) runs for around 35 pages, and the five hundred or so references there point in many directions. The weave of these elements differs greatly from citational patterns in the humanities or social sciences. Reading this book almost necessitates an archaeological approach since it comprises so many parts and fragments.

The citational fabric of *Elements of Statistical Learning* is woven with different threads, some reaching back into early twentieth statistics, some from post-WW2 cybernetics, many from information theory and then in the 1980s onwards, increasingly, from cognitive science and computer science (see figure 1.3 to get some sense of their distribution over time). While some of these references either point to Hastie, Tibshirani or Friedman's own publications, or that of their statistical colleagues, the references rove quite widely in other fields and over time. *Elements of Statistical Learning* as a text is processing, assimilating and recombining techniques, diagrams and data from many different places and times. (The different waves appearing in the references cited in (Hastie, Tibshirani, and Friedman 2009) will shape discussion in later chapters in certain ways; for instance, late 1990s biology is the topic of chapter ~~??~~ and optimization functions dating from the 1950s are discussed in chapter ~~??~~).

Both the inward and outward movements of citation suggest that *Elements of Statistical Learning*, like much in field of machine learning, has a matrix-like character that constantly superimposes and transposes elements across boundaries and barriers. The implication here is that machine learning as a knowledge practice has a highly interwoven texture, and in this respect differs somewhat from the classical understandings of scientific disciplines as bounded by communities of practice, norms and problems (as for instance, in Thomas Kuhn's account of normal science (Kuhn 1996). This aggregate or superimposed character of machine learning should definitely figure in any sense we make of it, and will ~~inevitably~~ affect how critical thought ~~might test itself against it~~. .

R: a matrix of transformations

Although barely a single line of code appears in *Elements of Statistical Learning*, all of the learners presented there are implemented in a single programming language, R. Coding is the operational practice that links the different planes and elements of machine learning in an operational formation. The authors ~~say~~ 'we used the R and S-PLUS programming languages in our courses' (Hastie, Tibshirani, and Friedman 2009, 9) but many elements of the book derive from R code.[4] The proliferation of

[4] In a later book by some of the same authors with the very similar sounding title *An Introduction to Statistical Machine Learning with Applications in R* (James et al. 2013), R does appear in abundance. This book, however, is much shorter and less inclusive in various ways. There are in any case many online manuals, guides and tutorials relating

programming languages such as FORTRAN (dating from the 1950s), C (1970s), C++ (1980s), then Perl (1990s), Java (1990s), Python (1990s) and R (1990s), and computational scripting environments such as Matlab, multiplied the paths along which machine learners move through operational formations. It would be difficult to comprehend the propagation of machine learners across domains of science, business and government without paying attention to coding practices. Even if textbooks and research articles are not read, software packages and libraries for machine learning are used. Code has a mobility that extends the diagrammatic practices of machine ~~learning~~ into a variety of settings and places where the scientific reading apparatuses of equations, statistical plots, and citations of research articles would not be operative.

How should we think of the R code in *Elements of Statistical Learning* in its operational specificity? Its growth is perhaps just as important as its operation (Mackenzie 2014). An open source programming language, according to surveys of business and scientific users, at the time of writing, R has replaced popular statistical software packages such as SPSS, SAS and

to R (Wikibooks 2013). For present purposes, I draw mainly on semi-popular books such as *R in a Nutshell* (Adler and Beyer 2010), *The Art of R Programming* (Matloff 2011), *R Cookbook* (Teetor 2011), *Machine Learning with R* (Lantz 2013) or *An Introduction to Statistical Learning with Applications in R* (James et al. 2013). These books are not written for academic audiences, although academics often write them and use them in their work. They are largely made up of illustrations and examples of how to do different things with different types of data, and their examples are typically oriented towards business or commercial settings, where, presumably, the bulk of the readers work, or aspire to work. Given a certain predisposition (that is, geekiness), these books and other learning materials can make for enjoyable reading. While they vary highly in quality, it is sometimes pleasing to see the economical way in which they solve common data problems. This genre of writing about programming specialises in posing a problem and solving it directly. In these settings, learning takes place largely through following or emulating what someone else has done with either a well known dataset (Fisher's iris) or a toy dataset generated for demonstration purposes. The many frictions, blockages and compromises that often affect data practice are largely occluded here in the interests of demonstrating the neat application of specific techniques. Yet are not those frictions, neat compromises and strains around data and machine learning precisely what we need to track diagrammatically? In order to demonstrate both the costs and benefits of approaching R through such materials, rather than through ethnographic observation of people using R, it will be necessary to stage encounters here with data that has not been completely transformed or cleaned in the interests of neatly demonstrating the power of a technique. One way to do this is by writing about code while coding.

	How often suggested
testthat	428
knitr	174
knitr, rmarkdown	168
testthat, knitr, rmarkdown	87
MASS	85
testthat, knitr	48
knitr, rmarkdown, testthat	47
RUnit	38
knitr, testthat	37
R.rsp	34
parallel	27
lattice	25
mvtnorm	18
testthat, covr	18
ggplot2	16
survival	14
rgl	12
testthat, covr, knitr, rmarkdown	10
akima	8
covr, testthat	8

Table 1.4: R packages suggested by the ElemStatLearn package

	Package	How often depended on
methods	methods	80
stats	stats	34
survival	survival	32
MASS	MASS	25
mvtnorm	mvtnorm	23
Matrix	Matrix	15
ggplot2	ggplot2	11
lattice	lattice	9
rJava	rJava	9
igraph	igraph	8
grid	grid	7
tcltk	tcltk	7
graphics	graphics	6
methods, stats	methods, stats	6
rgl	rgl	6
XML	XML	6
coda	coda	5
nlme	nlme	5
vegan	vegan	5
gtools	gtools	4

Table 1.5: 'R' packages depended on by the 'ElemStatLearn' package

Stata as the statistical and data analysis tool of choice for many people in business, government, and sciences ranging from political science to genomics, from quantitative finance to climatology (RexerAnalytics 2015). Developed in New Zealand in the mid-1990s, and like many open source software projects, emulating `S`, a commercialised predecessor developed at AT & T Bell Labs during the 1980s, `R` is now extremely widely used across life and physical sciences, as well as quantitative social sciences. John Chambers, the designer of `S`, was awarded the Association for Computing Machinery (ACM) 'Software System Award' in 1998 for 'the S system, which has forever altered how people analyze, visualize, and manipulate data' (ACM 2013). Many undergraduate and graduate students today earn `R` as a basic tool for statistics. Skills in `R` are often seen as essential pre-requisite for scientific researchers, especially in the life sciences. (In engineering, Matlab is widely used.) Research articles and textbooks in statistics commonly both use R to demonstrate methods and techniques, and create R packages to distribute the techniques and sample data. Nearly all of these publication-related software packages, including quite a few from the authors of *Elements of Statistical Learning* are soon or later available from the 'Comprehensive R Archive Network (CRAN)' (CRAN 2010). Estimates of its number of users range between 250000 and 2 million. Increasingly, R is integrated into commercial services and products (for instance, SAS, a widely used business data analysis system now has an R interface; Norman Nie, one of the original developers of the SPSS package heavily used in social sciences, now leads a business, Revolution, devoted to commercialising R; R is heavily used at Google, at FaceBook, and by quantitative traders in hedge funds; in 2013 'R usage is sky-rocketing'; etc.). In general terms, R has ~~a kind of disciplinary polyglot~~ currency as a form of expression, and exhibits a fine-grained relationality with many different epistemic and operational situations associated with machine learning.

Two early proponents of R and S describe the motivation for the language:

> The goal of the S language … is "to turn ideas into software, quickly and faithfully" … it is the duty of the responsible data analysts to engage in this process … the exercise of drafting an algorithm to the level of precision that programming requires can in itself clarify ideas and promote rigorous intellectual scrutiny. … Turning ideas into software in this way need not be an unpleasant duty. (Venables and Ripley 2002, 2)

Bill Venables and Brian Ripley, statisticians working on developing S, the almost identical commercial predecessor to R, wrote in the early 1990s of the responsibility of data analysts to write not just use software. They ~~write 'software'~~ here not in the sense of a product, but in the sense that today would more likely be called 'code.' Their sense of coding and programming as clarifying and concretising ideas with precision – as abstractions – has thoroughly taken hold in contemporary data analysis. If code, as they suggest, entails a threshold of idealisation, it differs from mathematical formalization in that it changes the positions and relations of knowledge to include machines, devices and infrastructures.

~~While t~~he view that code ~~is a precise expression of ideas~~ ~~makes sense, it~~ does not capture the relational complexity of R code as it operates in a setting such as *Elements of Statistical Learning*. ~~In machine learning, c~~ode, along with mathematics, is a primary operational form ~~that ideas take as they become~~ machine learners. But code as an expressive operation by which 'an individual formulates an idea' or as 'a rational activity that may operate in a system of inference' (Foucault 1972, 117) ~~does not exhaust and should not be conflated with~~ operational practice in machine learning. Similarly, the intersection of R with machine learning also lies somewhat at odds with Pedro Domingos' characterisation of machine learning as a shift away from people building to learners growing programs.

Listing ~~1~~3: Install packages in R

```
`install.packages('ElemStatLearn', dependencies='Suggests', repos
    ↪   = 'http://cran.us.r-project.org')
```

What in R (let alone other programming languages) overflows both the ideas of code as expression of ideas and code as automation? Alongside expression and automation, much R code ~~furnishes~~ a matrix of practice crossing network infrastructures, display screens, statistical techniques, software engineering architectures as well as publication and documentation standards. For instance, the line of R code shown in the listing ~~1~~3 when executed opens another way of reading *Elements of Statistical Learning* and ~~getting a feel for the dragnet~~ of practical relations and infrastructural configurations running through it. Take the part of the line `dependencies = 'Suggests'`. When the line of code executes, the stipulation of `dependencies` ~~leads to~~ a ~~quite~~ wide-ranging ~~installation event~~ that installs many `R` packages. If the installation works (and that assumes quite a lot of configuration work has already taken place; for instance, installing a recent version of the `R` platform), then *Elements of Statistical Learning* is now augmented by various pieces of code, and by various datasets that in some ways echo or mimic the book but in other ways extend it operationally (see tables ~~1~~ 5 and ~~1~~ 4).[5] These code elements are often stunningly specialised. As Karl Marx wrote of the 500 different hammers made in Birmingham, 'not only is each adapted to one particular process, but several varieties often serve exclusively for the different operations in one and the same process' (Marx 1986, 375~~) .~~ Something similar holds in `R`: thousands of software packages in online repositories suggest that a highly specialised division of labour ~~and possibly refined co-~~operative labour processes operate around data.

Because of this almost incoherent plurality, and its labile status as both a programming environment and a statistical analysis package, `R` is an evocative object, to use the psychoanalyst Christopher Bollas' term (Bollas 2008) , an object through which many different ways of thinking circulate. Standing somewhere at the intersection of statistics and computing, modelling and programming, many different disciplines, techniques, domains and actors

[5] Most of the packages associated with the `ElemStatLearn` implement methods or techniques developed by Hastie, Tibshirani or Friedman, but some are much more generic. `MASS` for instance is highly cited `R` package. (Of the 9857 packages in the R CRAN system, ~~0~~ depend on the library `MASS`, itself an adjunct to the influential and highly cited *Modern Applied Statistics with S* (Venables and Ripley 2002), a textbook that presents many machine learning techniques using `S` , AT & T Bell Labs commercial precursor to the open sourced `R`.) For our purposes, this hardly accidental mixing of academic or research work with a programming languages and its associated infrastructures is fortuitous. It allows us to transit between different strata of the social fields of science, engineering, health, medicine, business media and government more easily.

intersect in R. It engages immediately, practically and widely with words, numbers, images, symbols, signals, sensors, forms, instruments and above all abstract forms such as mathematical functions like probability distributions and many different architectural forms (vectors, matrices, arrays, etc.), as it employs data. If, as Bollas suggests, 'our encounter, engagement with, and sometimes our employment of, actual things is a *way* of thinking' (92), then it plausible that R not only gathers a plurality of data practices – working with measurements, numbers, text, images, models and equations, with techniques for sampling and sorting, with probability distributions and random numbers – but that it ~~embodies the kernel of a mode of thought relevant to contemporary realities.~~

The obdurate mathematical glint of machine learning

If scientific research literature and operational R code constitute the elements of an operational formation presented in *Elements of Statistical Learning*, what of the mathematics? While references from many different places flow in and out of *Elements of Statistical Learning*, they are nearly all articulated in mathematical form. Machine learning as a grouping of statements relies heavily on mathematics. Given that mathematics is itself diverse and multistranded, what kind of mathematics matters here? While later chapters will explore some of the main mathematical practices (linear algebra, statistical inference, etc.), machine learners in *Elements of Statistical Learning* coalesce around a single exemplary technique: linear regression models or fitting a line to points. The linear regression model is pivotal, not just in *Elements of Statistical Learning* but in much of the scientific and engineering literature. The linear regression model pushes up some of the citational peaks in Figure

~~1~~3. Even though it is an old technique dating back to Francis Galton in the 1890s (see (Stigler 1986, chapter 8)), it remains perhaps the central working element of machine learning.

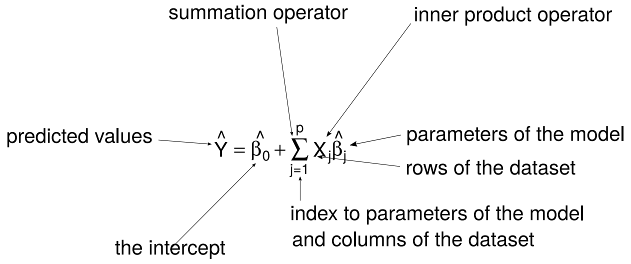
Elements of Statistical Learning acknowledges the statistical legacy and inheritance in machine learning:

> The linear model has been a mainstay of statistics for the past 30 years and remains one of our most important tools. Given a vector of inputs $X^T = (X_1, X_2, ..., X_p)$, we predict the output Y via the model
>
> $$\hat{Y} = \hat{\beta}_0 + \sum_{(}^{p} j = 1) X_j \hat{\beta_j})$$
>
> The term $\hat{\beta}_0$ is the intercept, also known as the *bias* in machine learning (Hastie, Tibshirani, and Friedman 2009, 11).

In the course of the book, this mathematical expression appears in many variations, iterations, expansions and modifications ('ridge regression'; 'least angle regression'; 'project pursuit,' etc.). But this introduction of the 'mainstay of statistics,' the linear model, already introduces a diagrammatic element – the mathematical equation – that is perhaps the most prominent feature in the text.

Any reading of the book has to work out a way to traverse the forms show in equation ~~??~~.

In its relatively compressed typographic weave, expressions such as Equation ~~1~~4 operationalize movements through data or 'a vector of inputs X^T.' These expressions, which are not comfortable reading for non-technical readers, ~~are however worth looking at carefully~~ if we want to move 'at the same level of abstraction as the algorithm' (Pasquinelli 2015). They can be found in hundreds in (Hastie, Tibshirani, and Friedman 2009), but also in many



$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j$$

Figure ~~1~~.4: The linear regression model other places. ~~While their~~ presence distinguishes machine learning from parts of computer science where mathematical equations are less common~~,~~ mathematical formalizations also allow the book to ~~suture a panoply of~~ scientific publications and datasets in fields of statistics, artificial intelligence and computer science. Along with the citations, the graphical plots, and the code relationality, these equations are integral connective tissue in machine ~~learning.~~

In dealing with the obscuring dazzle of mathematical formalisation, I find it useful here to follow Charles Sanders ~~Peirce account of~~ mathematics. 'Mathematical reasoning,' he writes, 'is diagrammatic' (Peirce 1998, 206). ~~For Peirce, we should see~~ mathematics, whether it takes an algebraic or geometrical form, whether it appears in symbols, letters, lines or curves as diagrams. For Peirce, a diagram is a kind of 'icon.' The icon is a sign that resembles the object it refers to: it has a relation of likeness. What likeness appears in equation ~~1~~4? ~~Peirce says: 'm~~any diagrams resemble their objects not all in their looks; it is only in respect to the relations of their parts that their likeness consists' (13). As we have seen in the perceptron code, machine learners can be expressed in statements in a programming language like `Python` or `R`. In code, however, the relations between parts cannot be observed in the same way as they can in the algebraic form. The 'very idea of the art,' as Peirce puts it (Peirce 1992, 228), of algebraic expressions is that the formulae can be manipulated, or that component elements can be moved without much effort through substitutions, transformations and variations. The graphic form of the expression include the various classical Greek operator symbols such as or , as well as the letters x, y, z and the indices (indexical signs) such as j that appear in subscript or superscript, as well as the spatial arrangement of all these elements in lines and sometimes arrays. A variety of relations run between these different symbols and spatial arrangements. For instance, in ~~all such expressions~~, the relation between the left hand side of the '=' and the right hand side is very important. By convention, the left hand side of the expression is the value that is predicted or calculated (the 'response' variable) and the right hand side are the input variables or 'features' that contribute data to the model or algorithm. This spatial arrangement fundamentally affects the design of algorithms. In the case of equations ~~??~~, the '^' over $\hat{Y}$ symbolises a predicted value rather than a value that can be known completely through deduction, derivation or calculation. This distinction between predicted and actual values organizes a panoply of different practices and imperatives (for instance, to investigate the disparities between the predicted and actual values – machine learning practitioners spend a lot of time on that problem).

The broad point is that the whole formulae is a diagram, or an icon that '*exhibits*, by means of the algebraical signs (which are not themselves icons), the relations of the quantities concerned' (Peirce 1998, 13). Because diagrams ~~suppress~~ so many details, they allow one to focus on a selected range of relations between parts. ~~This affordance of~~ diagrammatic mathematical forms is extremely important in the operational formation of machine learning. Diagrams can diagram other diagrams. Operations can themselves become the subject of operations. The nesting of diagram is ~~highly~~ generative ~~since~~ it allows what Peirce calls 'transformations' (212) or the construction of 'a new general predicate'(Peirce 1992, 303).[6]

[6] Félix Guattari makes direct use of Peirce's account of diagrams as icons of relation in his account of 'abstract machines' (Guattari 1984). He writes that 'diagrammaticism brings into play more or less de-territorialized trans-semiotic forces, systems of signs, of code, of catalysts and so on, that make it possible in various specific ways to cut across stratifications of every kind' (145). Here the 'trans-semiotic forces' include mathematical formulae and operations (such as the banking system of Renaissance Venice, Pisa and Genoa). They are trans-semiotic because they are not tethered by the signifying processes that code experience or speaking positions according to given stratifications such as class, gender, nation and so forth. While Guattari (and Deleuze in turn in their co-written works (Guattari and Deleuze 1988)) is strongly critical of the way which signification territorializes (we might think of cats patrolling, marking and displaying in order to maintain their territories), he is much more affirmative of diagrammatic processes. He

While I seek to relate to the ~~equations~~ as diagrams, and will present a selection of ~~them~~ (nowhere near as many as found in *Elements of Statistical Learning*) in the following pages, I am not assuming their operation is transparent or fully legible. Just as much as the analysis of a photograph, a literary work or an ethnographic observation, their diagrammatic composition calls for repeated consideration. Peirce advises not to begin with examples that are too simple: 'in simple cases, the essential features are often so nearly obliterated that they can only be discerned when one knows what to look for' (Peirce 1998, 206). He also suggests 'it is of great importance to return again and again to certain features' (206). Looking at these diagrammatic expressions repeatedly might allows us to map something of how transformations, generalisations or intensification flow across disciplinary boundaries, across social stratifications, and sometimes, generate potentially different ways of thinking about collectives, inclusion and belonging.

CS229, 2007: returning again and again to certain features

If we were to follow Peirce's injunction to 'return again and again to certain features,' how would we do that? *Elements of Statistical Learning* is a difficult book to read in isolation (although it does pay re-reading). Even after several years, the diagrammatic density of its 'elements' or statements (equations, citations, tables, datasets, plots) leaves me with a refractory feeling of 'not quite understanding.' This feeling is inevitable because the

calls them 'a-signifying' to highlight their difference from the signifying processes that order social strata. He suggests that diagrams become the foundation for 'abstract machines' and the 'simulation of physical machinic processes.' Writing in the 1960s, Guattari powerfully anticipates the abstract machines and their associated diagrams that have taken shape and physical form in the succeeding decades.

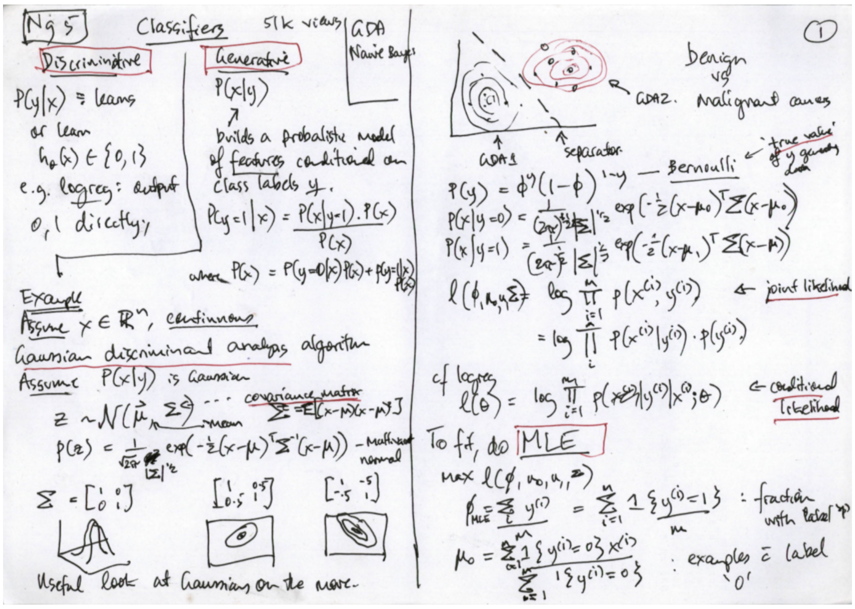
book condenses finished work from several disciplines, and partly because it seeks to organize a great diversity of materials *epistemically.* Indeed, the book might seen as evidence that machine learning has crossed an epistemic threshold formulated in a statistical apparatus. (The statistical aspects of machine learning are the main topic of chapter ~~??~~).

Does ~~a~~ computer science course offer ~~a more easily followed~~ route? Andrew Ng's course 'Machine Learning' CS229 at Stanford (http://cs229.stanford.edu/) might provide a supplementary path into machine learning (~~*Lecture 1 | Machine Learning (Stanford)*~~ 2008).[7] The course description runs as follows:

> This course provides a broad introduction to machine learning and statistical pattern recognition. Topics include supervised learning, unsupervised learning, learning theory, reinforcement learning and adaptive control. Recent applications of machine learning, such as to robotic control, data mining, autonomous navigation, bioinformatics, speech recognition, and text and web data processing are also discussed (~~*Lecture 1 | Machine Learning (Stanford)*~~ 2008)

CS229 is in many ways a typical computer science pedagogical exposition of machine learning. Machine learning expositions usually begin with simple datasets and the simplest possible statistical models and machine learners (linear regression), and then, with a greater or lesser degree of attention to issues of implementation, move through a succession of increasingly sophisticated and specialised techniques. This pattern is found in many of the how-to books, in the online courses, and in the academic textbooks, including (Hastie, Tibshirani, and Friedman 2009).

[7] A heavily shortened version of this course has been delivered under the title 'Machine Learning' on Coursera.org, a MOOC (Massive Open Online Course) platform.



Ng 5 Classifiers 51k views

GDA
Naive Bayes

Discriminative

$P(y|x)$ ⇒ learns
or learn
$h_\theta(x) \in \{0,1\}$
e.g. logreg: output
0, 1 directly,

Generative

$P(x|y)$
↗
builds a probalistic model
of features conditional on
class labels y.

$P(y=1|x) = \frac{P(x|y=1) . P(x)}{P(x)}$

where $P(x) = P(y=0|x)P(x) + \frac{P(y=1|x)}{P(x)}$

Example
Assume $x \in \mathbb{R}^n$, continuous
Gaussian discriminant analysis algorithm
Assume $P(x|y)$ is Gaussian

$z \sim \mathcal{N}(\vec{\mu}, \Sigma)$ . . . covariance matrix
mean $\Sigma = E[(x-\mu)(x-\mu)^T]$

$p(z) = \frac{1}{\sqrt{2\pi}\,|\Sigma|^{1/2}} \exp(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu))$ – Multivariate normal

$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ $\begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$ $\begin{bmatrix} 1 & -.5 \\ -.5 & 1 \end{bmatrix}$

Useful look at Gaussians on the move.

①

GDA1 separator GDA2.

Benign vs Malignant cancers

$P(y) = \phi^y (1-\phi)^{1-y}$ — Bernoulli ← 'true value' of y given data

$P(x|y=0) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp(-\frac{1}{2}(x-\mu_0)^T \Sigma (x-\mu_0))$

$P(x|y=1) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp(-\frac{1}{2}(x-\mu_1)^T \Sigma (x-\mu))$

$\ell(\phi, \mu_0, \mu_1 \Sigma) = \log \prod_{i=1}^{m} p(x^{(i)}, y^{(i)})$ ← joint likelihood

$= \log \prod_{i} p(x^{(i)}|y^{(i)}) \cdot p(y^{(i)})$

cf logreg
$\ell(\theta) = \log \prod_{i=1}^{m} p(y^{(i)}|x^{(i)}; \theta)$ ← conditional likelihood

To fit, do $\boxed{\text{MLE}}$

$\max \ell(\phi, \mu_0, \mu_1, \Sigma)$

$\phi_{MLE} = \frac{\sum_i y^{(i)}}{m} = \sum_{i=1}^{m} \frac{1\{y^{(i)}=1\}}{m}$: fraction with label '1'

$\mu_0 = \frac{\sum_{i=1}^{m} 1\{y^{(i)}=0\} x^{(i)}}{\sum_{i=1}^{m} 1\{y^{(i)}=0\}}$: examples with label '0'

Figure ~~1.~~5: Class notes lecture 5, Stanford CS229, 2007

Ng's CS229 lectures differ from most other pedagogical materials in that we see someone writing and deriving line after line of equations using chalk on a blackboard. Occasionally, questions come from students in the audience (not shown on the Youtube videos), but mostly Ng's transcription of equations and other diagrams from the paper notes he holds to blackboard continues uninterrupted.[8] Ng's Youtube anachronistic but popular lectures have a certain diagrammatic density that comes from the many hours of ~~chalked writing he generates~~ during the course deriving equations. In a time when PowerPoint presentations or some other electronic textuality would very much have been the norm (2007), ~~why~~ is a computer science professor, teaching a fairly advanced postgraduate course, ~~writing~~ on a chalkboard by hand?

[8] After sitting through 20 hours of Ng's online lectures, and attempting some of the review questions and programming exercises, including implementing well-known algorithms using `R`, one comes to know datasets such as the San Francisco house price dataset and Fisher's `iris` (Fisher 1936) quite well. Like the textbook problems that the historian of science Thomas Kuhn long ago described as one of the anchor points in scientific cultures (Kuhn 1996), these iconic datasets provide an entry point to the 'disciplinary matrix' of machine learning. Through them, one gains some sense of how predictive models are constructed, and what kinds of algorithmic architectures and forms of data are preferred in machine learning.

Figure ~~1~~.5 shows a brief portion of around 100 pages of notes I made on this course. The act of writing down these equations and copying the many hand-drawn graphs Ng produced was a deliberative diagrammatic experiment in 'returning again and again' to what is perhaps overly hardened in *Elements of Statistical Learning*. Like the 50,000 or so other people who had watched this video, I partly complied with Ng's injunction to 'copy it, write it out, cover it, and see if you can reproduce it' (~~*Lecture 10 | Machine Learning (Stanford)* 2008~~). While it occasions much writing and drawing, and many struggles to keep up with the transformations and substitutions that Ng narrates as he writes, ~~it seems to me that writing out~~ derivations, with all their substitutions and variations, alongside the graphic sketches of intuitions about the machine learners, accesses something of the *diagrammatic composition* of machine learning that is quite hard to ~~track in~~ *Elements of Statistical Learning*. There the diagrammatic weave between the expressions of linear algebra, calculus, statistics, and the off-stage implementation in code is almost too tight to work with. In Ng's CS229 lectures, by contrast, the weaving, while still complex, is much more open. The lectures lack the citational tapestry of *Elements of Statistical Learning*. They are not able to wield the datasets and the panoply of graphic forms found there, and virtually no machine learning code appears on the blackboard (although the CS229 student assignments, also to be found online, are code implementations of the algorithms and models). ~~But~~ Ng's lectures move more slowly, and we begin to see some of the different groupings and associations comprising the operational formation. More importantly perhaps, this absorbing process of writing derivations might begin to transform ways of thinking, saying and knowing.

The visible learning of machine learning

For a book with 'learning' in its title, *Elements of Statistical Learning* has visibly little to say about how to learn machine learning. 'Learning' is briefly discussed on the first page of *Elements of Statistical Learning*, but the book hardly ever returns to the topic or even that term explicitly. We learn on page 2 that a 'learner' in machine learning is a model that predicts outcomes. (As I discuss in chapter ~~**??**~~, learning is comprehensively understood in machine learning as finding a mathematical *function* that could have generated the data, and optimising the search for that function as much as possible.) The notion of learning in machine learning derives from the field of artificial intelligence. The broad project of artificial intelligence, at least as envisaged in its 1960s-1970s heyday as a form of symbolic reasoning, is today largely regarded as a dead-end.

How then did learning get into the title of *Elements of Statistical Learning*? Does 'learning' anthropomorphize statistical modelling or computer programming? The so-called 'learning problem' and the theory of learning machines developed by researchers in the 1960-1970s was largely based on work already done in the 1950s on cybernetic devices such as the perceptron, the prototypical neural network model developed by the psychologist Frank Rosenblatt in the 1950s (Rosenblatt 1958). Drawing on the McCulloch-Pitts model of the neurone and mathematical techniques of optimization (Bellman 1961), Rosenblatt implemented the perceptron, which today would be called a single-layer neural network (Hastie, Tibshirani, and Friedman 2009, 393) as an electro-mechanical device at the Cornell University Aeronautical Laboratory in 1957. For present purposes, it is interesting to see what diagrams Rosenblatt used and how they differ from contemporary diagrams.

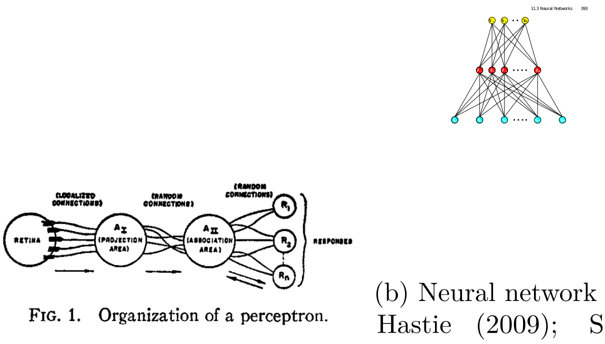


FIG. 1. Organization of a perceptron.

(a) The neurological perceptron (Rosenblatt, 1958, 389)

(b) Neural network from Hastie (2009); Single layer feedforward artificial neural network.

Figure ~~1~~.6: 1958 perceptron and 2001 neural net compared

If we compare the diagram of Rosenblatt's perceptron shown in Figure ~~1~~.6a to the typical contemporary diagram of a neural network shown in Figure ~~1~~.6b, the differences are not that great in many ways. The diagram of a neural network found in Rosenblatt's paper (Rosenblatt 1958) has no mathematical symbols on it, but the one from (Hastie, Tibshirani, and Friedman 2009) does. Rosenblatt's retains neuro-cognitive-anatomical reference points (retina, association area, projection area) whereas Hastie et. al.'s replaces them with the symbols that we have already seen in play in the expression for the linear model ~~??~~. What has happened between the two diagrams? As Vladimir Vapnik, a leading machine learning theorist, observes: 'the perceptron was constructed to solve pattern recognition problems; in the simplest case this is the problem of constructing a rule for separating data of two different categories using given examples' (Vapnik 1999, 2). While computer scientists in artificial intelligence of the time, such as Marvin Minsky and Seymour Papert, were sceptical about the capacity of the perceptron model to distinguish or 'learn' different patterns (Minsky and Papert 1969), later work showed that perceptrons could 'learn universally'[9].

[9] In describing the entwined elements of machine learning techniques, and citing

For present purposes, the key point is not that neural networks have turned out several decades later to be extremely powerful algorithms in learning to distinguish patterns, and that intense research in neural networks has led to their ongoing development and increasing sophistication in many 'real world' applications (see for instance, for their use in sciences (Hinton and Salakhutdinov 2006), or in commercial applications such as drug prediction (Dahl 2013) and above all in the current surge of interest in 'deep learning' by social media platform and search engines such as Facebook and Google). For our purposes, the important point is that the notion of the learning machine ~~began to establish~~ an ongoing diagonalization that transforms ~~basic diagrammatic pattern~~ through substitutions of increasingly convoluted or nested operations. The whole claim that machines 'learn' rests on ~~this diagrammatization~~ that recurrently and sometimes recursively transforms

various machine learning textbooks, I'm not attempting to provide any detailed history of their development. My archaeology of these developments does not explore the archives of institutions, laboratories or companies where these techniques took shape. It derives much more either from following citations back out of the highly distilled textbooks into the teeming collective labour of research on machine learning as published in hundreds of thousands of articles in science and engineering journals, or from looking at, experimenting with and implementing techniques in code. For instance, (Olazaran 1996) offers a history of the perceptron controversy from a science studies perspective. During the 1980s, artificial intelligence and associated approaches (expert systems, automated decision support, neural networks, etc.) were a matter of some debate in the sociology and anthropology of science. The work of Harry Collins would be one example of this (Collins 1990), Paul Edwards on artificial intelligence and Cold War (Edwards 1996), Nathan Ensmenger on chess (Ensmenger 2012), and Lucy Suchman on plans and expert systems (L. A. Suchman 1987) would be others. Philosophers such as Hubert Dreyfus (*What Computers Can't Do* (Dreyfus 1972, 1992) had already extensively criticised AI. In the 1990s, the appearance of new forms of simulation, computational fields such as a-life and new forms of robotics such as Rodney Brook's much more insect-like robots at MIT attracted the interest of social scientists (Helmreich 2000) amongst many others. Sometimes this interest was critical of claims to expertise (Collins), and at other times, interested in how to make sense of the claims without foreclosing their potential novelty (Helmreich). By and large, I don't attend to controversies in machine learning as a scientific field, and I don't directly contest the epistemic authority of the proponents of the techniques. I share with Lucy Suchman an interest in how the 'effect of machine-as-agent is generated' and in how 'translations ... render former objects as emergent subjects'(L. Suchman 2006, 2). I diverge around the site of empirical attention. I'm persevering with the diagrams in each of the following chapters in order to track the movement of tendencies that are not so visible in terms of either the controversies or the assumptions of agency embodied in many AI systems of the 1980s. The agency of machine learning, in short, might not reside so much in any putative predictive or classificatory power if manifests, but rather its capacity to mutate and migrate across contexts.

the icon of relations, sometimes in the graphic forms shown above ~~and~~ more often in the algebraic patterns.

The changes in graphics suggest transformations in the operational formation. I am suggesting, then, that we should follow the transformations associated with machine learning diagrammatically, provided we maintain a rich understanding of diagram, and remain open to multiple vectors of abstraction. Following Peirce, we might begin to see machine learning as a diagrammatic practice in which different semiotic forms – lines, numbers, symbols, operators, patches of colour, words, images, marks such as dots, crosses, ticks and arrowheads – are constantly connected, substituted, embedded or created from existing diagrams. The diagrams we have already seen from *Elements of Statistical Learning* - algebraic formulae and network topology - don't exhaust the variations at all. Just a brief glance through this book or almost any other in the field shows not only many formulae, but tables, matrices, arrays, line graphs, contour plots, scatter plots, dendrograms and trees, as well as algorithms expressed as pseudo-code. The connections between these diagrams are not always very tight or close. Learning to machine learning (whether you are a human learner or a learner in the sense of a machine) means ~~dancing~~ between diagrams. ~~This dance is~~ relatively ~~silent~~ and sometimes almost ~~motionless~~ as signs slide between different diagrammatic articulations. ~~Diagrammatization offers then a way to track the ongoing project which tries treat data like farmers treat crops (see epigraph from Domingos in this chapter).~~ To understand what machines can learn, we need to look at how they have been drawn, designed, or formalised. But what in this work of designing and formalising predictive models is like farming? Some very divergent trajectories open up here. On the one hand, the diagrams become machines when they are implemented. On the other hand, the machines generate new diagrams when they function.

We need to countenance both forms of movement in order to understand any of the preceding diagrams – the algebraic expressions or the diagrams of models such as the perceptron or its descendants, the neural network. This means going downstream from the textbooks into actual implementations and places where people, algorithms, and machines mingle more than they do in the relatively ~~neat formality~~ of the textbooks. Rather than history or controversies in the field, I focus on the migratory patterns of methods, and the many configurational shifts associated with their implementations as the same things appears in different places.

The diagram of an operational formation

Does machine learning radically change programming practice? Programmability does change but only through gyrations between epistemic, infrastructural and discursive heterogeneities practised and practised in code. I have been suggesting that we can get a sense of the heterogeneities and regularities of machine learning by treating *Elements of Statistical Learning* as a diagram that links and indexes the operations of machine learners in publication, in computation, in code. Mapped through research publications, pedagogical materials or code libraries in `R`, these operations form a primary field of expressions issuing from many parts. These parts include the accumulation or positivity of scientific literature with all ~~its~~ referentiality. They include mathematical derivation and formalization as an ~~accelerated~~ diagrammatic movement. The parts ~~include code as materially~~ weaving of infrastructures, conventions, standards, techniques, devices and collective relations. Statements (especially linear algebra, calculus, statistics), problems and techniques from multiple scientific disciplines (especially computer science, but also biology, medicine and others), devices such as computing platforms, data formats, and code repositories populate the operational formation. The operational power of machine learning does not stem from a single layer of abstraction. ~~The~~ diagrammatic forms of movement we have begun to discern in the polymorphic *Elements of Statistical Learning* ~~suggest key lines and paths worth following in opening up that engagement~~. Like the perceptron calculating weights that allow it to express the logic of the NAND function, we might first of all ~~turn to~~ the table, the ordering and aligning of numbers on which nearly all machine learning depends.

Glossary

$\hat{\beta}$ is a commonly used symbol for the model parameters, weights or coefficients. Estimating optimum values of β is a preoccupation in machine learning.

$\sum$ is an operator that sums together all the terms to the right of the symbol.

archaeology Michel Foucault defines archaeology as a description that explores the production of statements at the level of knowledge practices (*savoir*). It emphasizes the irregularities and discontinuities in knowledge practices as well as the derivations of operations and functions.

bias of a model refers to its inevitable approximation and misalignment to the actual processes that generated the data.

classifier is a machine learner that assigns instances to classes or categories such as `survive` or `die`, `cat` or `dog`.

cost function is a function that measures the difference between the output of the model (the prediction) and the known values (Whitehead,1960).

cross-validate is an operation that validates a model against a part of the data in order to gauge how well predictions generalize to fresh or hitherto unseen data. Many rounds of cross-validation may be used in training models when data is limited..

data strain borrows from A.N. Whitehead's notion of strain, which refers to implicit forces or tensions in bodies of data that relate to the feeling of geometrically straight or flat loci.

decision boundary is a boundary or surface drawn in vector space by a machine learning classifier to differentiate or separate and hence classify cases.

deep learning a neural network comprising many layers commonly used for image recognition.

diagram is a form of abstraction concerned with functioning and operations. In Gilles Deleuze's reading of Michel Foucault, diagrams display relations of force, and construct models of truth (Deleuze,1994).

discourse For Michel Foucault, a discourse groups statements generated by an enunciative function.

enunciative function For Michel Foucault, the mapping of statements to themselves, to subject positions, to correlate domains and their material forms of reuse, replication and transcription together generate statements. In this book, the many predictions, inferences, plots, tabulations, numbers, scores, probabilities, classifications, software libraries and devices comprise the enunciative function of machine learning.

enunciative modality For Michel Foucault, the sites, forms of observing, describing, teaching, perceiving associated with statements.

feature Also known in machine learning as variable, measurement, observation or attribute, a feature occupies one dimension in the vector space inhabited by data.

function Mathematically, a function uniquely maps one set of numbers onto another set of numbers. each other. In machine learning, functions operate diversely, sometimes transforming data to generate feature or vector spaces, sometimes measuring cost or loss for particular models, and sometimes expressing forms such as curves and surfaces that transform data. Across these different usages and domains, the operation of mapping or relation between sets of values such as X and Y can be seen.

generative model uses probability distributions to model the process that generated the data, thus allowing the model to generate or simulate samples from the data.

machine learner refers to humans and machines involved in learning from data together.

operational formation is a variation on Michel Foucault's discursive formation that highlights the collective human-machine regularities of power-knowledge. While operation and operational fields are implicit to discursive practice, they are somewhat overshadowed by the figures of the document, the utterance, and the proposition in Foucault's account.

partial derivative is an operator from differential calculus that expresses the rate of change of one variable with respect to another.

partial observer in Gilles Deleuze and Félix Guattari's concept of what a mathematical function does in science (Deleuze,1994).

perceptron A machine learner developed in the 1950s by Frank Rosenblatt. It is modelled on a neurone that learns to classify the input data or what it 'perceives' by varying parameters or weights on the sum of its inputs to produce values of either '1' or 0'.

positivity Michel Foucault's term in *Archaeology of Knowledge* to describe the specific forms of accumulation of a group of statements in a discursive formation.

probabilization The process where machine learners themselves are constituted as a population of devices, whose distribution and tendencies can be treated statistically.

referential : for Michel Foucault, the referential of a statement is not the referent (the facts, things, realities or beings designated) but the place, condition, field of emergence or principle of differentiation for the entities named, described or designated in the statement. The referentials for machine learning include various hyperobjects such as genomes, social media, epidemics, markets and economies. Such referentials encompass many named entities.

regularization operates on the referentials of machine learning to target subtle, diffuse distributions of difference in order to classify, estimate and rank their effects.

statement Michel Foucault's term for the product of an enunciative function that operationally relates a number of elements to a field of objects, establishing subject positions associated with them, and configuring a domain of coordination in which these elements can be invoked, used, and repeated. Statements take many forms including utterances, graphs, equations and numbers (Foucault,82).

variance of a model refers to its dependence on the particular data it is trained on.

vector Three senses of the term are relevant: 1. A vector as an element of vector space; 2. A data structure in programming languages such as `R` – a one dimensional array of elements; 3. A feeling in the sense used by A.N. Whitehead to describe the transfer from 'there' to 'here'.

vector space is a hyperspace of indefinite dimensions generated by the projective mapping of data variables or features into distinct coordinate dimensions.

vectorize operations on data that transform vectors of values in aggregate.

Bibliography

ACM. 2013. "John M. Chambers - Award Winner." Accessed December 12, 2013. http://awards.acm.org/award_winners/chambers_6640862.cfm.

Adler, Joseph, and Jörg Beyer. 2010. *R in a Nutshell.* O'Reilly Germany.

Alpaydin, Ethem. 2010. *Introduction to Machine Learning.* Cambridge, Massachusetts; London: MIT Press.

Bellman, Richard. 1961. *Adaptive Control Processes: A Guided Tour.* Vol. 4. Princeton N.J.: Princeton University Press.

Bishop, Christopher M. 2006. *Pattern Recognition and Machine Learning.* Vol. 1. New York: Springer.

Bollas, Christopher. 2008. *The Evocative Object World.* London & New York: Routledge.

Campbell-Kelly, Martin. 2003. *From Airline Reservations to Sonic the Hedgehog: A History of the Software Industry.* Cambridge, MA: MIT Press.

Coleman, Gabriella. 2012. *Coding Freedom: The Ethics and Aesthetics of Hacking.* Princeton N.J.: Princeton University Press.

Collins, Harry M. 1990. *Artificial Experts: Social Knowledge and Intelligent Machines.* Inside technology. Cambridge, MA.: MIT Press.

Conway, Drew, and John Myles White. 2012. *Machine Learning for Hackers.* Sebastopol, CA: O'Reilly.

Cox, Geoff. 2012. *Speaking Code: Coding as Aesthetic and Political Expression.* MIT Press.

CRAN. 2010. "The Comprehensive R Archive Network." Accessed May 5, 2010. http://www.stats.bris.ac.uk/R/.

Dahl, George. 2013. "Deep Learning How I Did It: Merck 1st Place Interview." Accessed June 17, 2013. http://blog.kaggle.com/2012/11/01/deep-learning-how-i-did-it-merck-1st-place-interview/.

Deleuze, Gilles. 1988. *Foucault.* Translated by Seân Hand. Minneapolis: University of Minnesota Press.

Domingos, Pedro. 2012. "A Few Useful Things to Know about Machine Learning." *Communications of the ACM* 55 (10): 78–87.

———. 2015. *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World.* New York: Basic Civitas Books.

Dreyfus, Hubert L. 1972. *What Computers Can't Do.* New York: Harper & Row.

———. 1992. *What Computers Still Can't Do: A Critique of Artificial Reason.* Cambridge, MA: MIT Press.

Edwards, Paul N. 1996. *The Closed World : Computers and the Politics of Discourse in Cold War.* Inside technology. Cambridge, MA; London: MIT Press.

Ensmenger, Nathan. 2012. "Is Chess the Drosophila of Artificial Intelligence? A Social History of an Algorithm." *Social Studies of Science* 42, no. 1 (): 5–30.

Fisher, Ronald A. 1936. "The Use of Multiple Measurements in Taxonomic Problems." *Annals of eugenics* 7 (2): 179–188.

Flach, Peter. 2012. *Machine Learning: The Art and Science of Algorithms That Make Sense of Data.* Cambridge University Press.

Foucault, Michel. 1972. *The Archaeology of Knowledge and the Discourse on Language.* Translated by Allan Sheridan-Smith. New York: Pantheon Books.

Gruber, John. 2004. "Markdown: Syntax." Accessed July 1, 2013. http://daringfireball.net/projects/markdown/.

Guattari, Félix. 1984. *Molecular Revolution : Psychiatry and Politics.* Harmondsworth, Middlesex, England ; New York, N.Y., U.S.A.: Penguin.

Guattari, Felix, and Gilles Deleuze. 1988. *A Thousand Plateaus: Capitalism and Schizophrenia.* London: Athlone, 1988.

Hastie, Trevor, Robert Tibshirani, and Jerome H. Friedman. 2001. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* 1st edition. New York: Springer.

———. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* 2nd edition. New York: Springer.

Helmreich, Stefan. 2000. *Silicon Second Nature : Culturing Artificial Life in a Digital World.* Berkeley, Calif. ; London: University of California Press.

Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. 2006. “Reducing the Dimensionality of Data with Neural Networks.” *Science* 313 (5786): 504–507.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An Introduction to Statistical Learning.* Springer.

Kirk, Matthew. 2014. *Thoughtful Machine Learning: A Test-Driven Approach.* 1 edition. Sebastopol, Calif.: O’Reilly Media.

Kuhn, Thomas S. 1996. *The Structure of Scientific Revolutions.* Chicago, IL: University of Chicago Press.

Lamport, Leslie, and A. LaTEX. 1986. *Document Preparation System.* Reading, MA: Addison-Wesley.

Lantz, Brett. 2013. *Machine Learning with R.* Birmingham: Packt Publishing.

Lecture 1 | Machine Learning (Stanford). 2008.

Lecture 10 | Machine Learning (Stanford). 2008. In collaboration with Andrew Ng.

Lee, D. D., and H. S. Seung. 1999. “Learning the Parts of Objects by Non-Negative Matrix Factorization.” *Nature* 401, no. 6755 (): 788–791. pmid: 10548103.

Mackenzie, Adrian. 2006. *Cutting Code: Software and Sociality.* Digital Formations. New York: Peter Lang.

———. 2014. “UseR! Aggression, Alterity and Unbound Affects in Statistical Programming.” In *Fun and Software: Exploring Pleasure, Paradox and Pain in Computing,* edited by Olga Goriunova. New York: Bloomsbury Academic.

Malley, James D., Karen G. Malley, and Sinisa Pajevic. 2011. *Statistical Learning for Biomedical Data.* 1st ed. Cambridge University Press.

Marx, Karl. 1986. *Capital A Critique of Political Economy. The Process of Production of Capital.* Moscow: Progress.

Matloff, Norman S. 2011. *Art of R Programming.* San Francisco: No Starch Press.

Minsky, Marvin, and Seymour Papert. 1969. “Perceptron: An Introduction to Computational Geometry.” *The MIT Press, Cambridge, expanded edition* 19:88.

Mitchell, Tom M. 1997. *Machine Learning.* New York, NY [u.a.: McGraw-Hill.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. “Playing Atari with Deep Reinforcement Learning” (). arXiv: 1312.5602 [cs].

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, et al. 2015. “Human-Level Control through Deep Reinforcement Learning.” *Nature* 518, no. 7540 (): 529–533.

Mol, Annemarie. 2003. *The Body Multiple: Ontology in Medical Practice.* Durham, N.C: Duke University Press.

Montfort, Nick, and Ian Bogost. 2009. *Racing the Beam: The Atari Video Computer System.* MIT Press.

Morton, Timothy. 2013. *Hyperobjects: Philosophy and Ecology After the End of the World.* Univ Of Minnesota Press.

Muenchen, Robert A. 2014. "The Popularity of Data Analysis Software." Accessed September 2, 2015. http://r4stats.com/articles/popularity/.

Olazaran, Mikel. 1996. "A Sociological Study of the Official History of the Perceptrons Controversy." *Social Studies of Science* 26, no. 3 (): 611–659.

Pasquinelli, Matteo. 2015. "Anomaly Detection: The Mathematization of the Abnormal in the Metadata Society." Berlin.

Pearson, Karl. 1901. "LIII. On Lines and Planes of Closest Fit to Systems of Points in Space." *Philosophical Magazine Series 6* 2, no. 11 (): 559–572.

Peirce, Charles Sanders. 1992. *The Essential Peirce: 1867-1893 v. 1: Selected Philosophical Writings.* John Wiley & Sons.

———. 1998. *The Essential Peirce - Volume 2: Selected Philosophical Writings: (1893-1913) v. 2.* Indiana University Press.

Perez, Fernando, and Brian E. Granger. 2007. "IPython. A System for Interactive Scientific Computing." *Computing in Science & Engineering* 9 (3): 21–29.

RexerAnalytics. 2015. "Rexer Analytics 7th Annual Data Miner Survey - 2015." Accessed May 9, 2011. http://www.rexeranalytics.com/Data-Miner-Survey-2015-Intro.html.

Ripley, Brian. 1996. *Pattern Recognition and Neural Networks. 1996.* Cambridge ; New York: Cambridge University Press.

Rosenblatt, F. 1958. "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain." *Psychological Review* 65 (6): 386–408.

Russell, Matthew A. 2011. *Mining the Social Web.* Sebastopol, CA: O'Reilly.

Schutt, Rachel, and Cathy O'Neil. 2013. *Doing Data Science.* Sebastopol, Calif.: O'Reilly & Associates Inc.

Segaran, Toby. 2007. *Programming Collective Intelligence: Building Smart Web 2.0 Applications.* Sebastapol CA.: O'Reilly.

Stigler, Stephen M. 1986. *The History of Statistics: The Measurement of Uncertainty before 1900.* Cambridge, Mass.: Harvard University Press.

Suchman, Lucy. 2006. *Human and Machine Reconfigurations: Plans and Situated Actions.* 2nd ed. Cambridge University Press.

Suchman, Lucy A. 1987. *Plans and Situated Actions : The Problem of Human-Machine Communication.* Cambridge: Cambridge University Press.

Teetor, Paul. 2011. *R Cookbook.* O'Reilly Media, Incorporated.

Vapnik, Vladimir. 1999. *The Nature of Statistical Learning Theory.* 2nd ed. 2000. Springer.

Venables, William N., and Brian D. Ripley. 2002. *Modern Applied Statistics with S.* Springer.

What We're Learning from Online Education | Video on TED.com. 2012. In collaboration with Daphne Koller.

Whitehead, Alfred North. 1956. *Modes of Thought; Six Lectures Delivered in Wellesley College, Massachusetts, and Two Lectures in the University of Chicago.* New York, Cambridge University Press.

Wikibooks. 2013. "R Programming - Wikibooks, Open Books for an Open World." Accessed June 27, 2013. http://en.wikibooks.org/wiki/R_Programming.

Perceptron. 2013. In *Wikipedia, the Free Encyclopedia,* by Wikipedia.

Xie, Yihui. 2013. "Knitr: A General-Purpose Package for Dynamic Report Generation in R." *R package version* 1.

Xie, Yihui, and J. J. Allaire. 2012. "New Tools for Reproducible Research with R." Warwick, UK.

Index