tiny [table]font=tiny

vectorisedname=vectorised, description=Operations on data that transform vectors of values.,see=[see also]vector

Machine Learning: Archaeology of a Data Practice

Adrian Mackenzie

Preface

This book is not an ethnography although it has an ethnographic situation. If it has a field site, it lies close to the places where the writing was done – in universities, on campuses, in classrooms and online training courses (including MOOCs), and then amidst the books, documents, websites, software manuals and documentation, and a rather vast accumulation of scientific publications.

Readers familiar with textbooks in computer science and statistics can see the traces of this site in some typographic conventions drawn from the fields I write about. Important conventions include:

1. Typesetting the name of any actual machine learners, code or devices that do machine learning, and datasets on which machine learners operate in a san serif font like `this`;
2. Presenting formulae, functions, and equations using the bristling indexicality of mathematical typography

Why emulate the apparatus of science and engineering publication in this way? Social science and humanities researchers, even when they are observant participants in their field sites, rarely experience a coincidence between their own writing practices and those of research subjects. The object of study in this book is a knowledge practice that documents itself in code, equations, diagrams and statements circulated in articles, books and various online formats (blogs, wikis, software repositories).

A dense feedback signal runs through the many propositions, formulations, diagrams, equations, citations and images in this book. I've been writing code for years (Mackenzie 2006). Writing code was nearly always something distant from writing about code. Recent developments in ways of analysing and publishing scientific data bring coding and writing closer together, such that implementing things can be done almost in the same space as writing about those things. This looping between writing code and writing about code brings about sometimes generative, sometimes frustrating, encounters with various scientific knowledge (mathematics, statistics, computer science), with infrastructures on many scales (ranging across networks, databases here and there, hardware and platforms of various kinds, as well as interfaces) and many domains. At many points in researching the book, I digressed a long way into quite technical domains of statistical inference, probability theory, linear algebra, dynamic models as well as database design and data standards. Much of the code I've written in implementing machine learning models or in reconstructing certain data practices does not appear in this text, just as not all of the words I've written in trying to construct arguments or think about data practices has been included. Much has been cut away and left on the ground (although the `git` repository of the book preserves many traces of the writing and code; see https://github.com/datapractice/machinelearning). So, like the recipe books, cookbooks, how-tos, tutorials and other documentations I have read, the code, the graphics and the prose have been tidied here. Many exploratory forays are lost and almost forgotten. Nevertheless, the several years I have spent writing about data practice has felt substantially different to any other project by virtue of a strong coupling between code in text, and text in code. Practically, this is made possible by working on code and text within the same file, in the same text editor. Switching between writing `R` and Python code (about which I say more below) to retrieve data, to transform it, to produce graphics, to construct models or some kind of graphic image, and within the same file be writing academic prose, might be one way to write about machine learning as a data practice.

programming languages!as mode of writing

The capacity to mix text, code and images depends on an ensemble of software tools that differ somewhat from the typical social scientist or humanities researchers' software toolkit of word processor, bibliographic software, image editor and web browser. In particular, it relies on software packages in the `R` programming language such as the '`knitr`' (Xie 2013; Xie and Allaire 2012) and in python, the `ipython` notebook environment (Perez and Granger 2007). Both have been developed by scientists and statisticians in the name of 'reproducible research.' Many examples of this form of writing can be found on the web: see IPython Notebook Viewer for a sample of these. These packages are designed to allow a combination of code written in `R`, python or other programming languages, scientific text (including mathematical formula) and images to be included, and importantly, executed together. In order to do this, they typically combine some form of text formatting or 'markup,' that ranges from very simple formatting conventions (for instance, the 'Markdown' format used in this book is much less complicated than HTML, and uses markup conventions readable as plain text and modelled on email (Gruber 2004);) to the highly technical (LaTeX, the de-facto scientific publishing format or 'document preparation system' (Lamport and LaTEX 1986) elements of which are also used here to convey mathematical expressions). They add to that blocks of code and inline code fragments that are executed as the text is formatted in order to produce results that are shown in the text or inserted as figures in the text.[1]

1. There are a few different ways of weaving together text, computation and images together. Each suffers from different limitations. In `ipython`, a scientific computing platform dating from 2005 (Perez and Granger 2007) and used across a range of scientific settings, interactive visualization and plotting, as well as access to operating system functions are brought together in a `Python` programming environment. Especially in using the `ipython` notebook, where editing text and editing code is all done in the same window, and the results of changes to code can be seen immediately, practices of working with data can be directly woven together with writing about practice. By contrast, `knitr` generates documents by combining text passages and the results (graphs, calculations, tabulations of data) of code interleaved between the text into one output document. When `knitr` runs, it executes the code and inserts the results (calculations, text, images)

In making use of the equipment created by the people I study, I've attempted to bring the writing of code and writing about code-like operations into proximity. Does proximity or mixing of writing code and writing words make a practical difference to an account of practice? If recent theories of code and software as forms of speech, expression or performative utterance are right (Cox 2012; Coleman 2012), it should. Weaving code through writing in one domain of contemporary technical practice, machine learning, might be one way of keeping multiple practices present, developing a concrete sense of abstraction and allowing an affective expansion in relation to machines.

in the flow of text. Practically, this means that the text editor used to write code and text, remains somewhat separate from the software that executes the code. By contrast, `ipython` combines text and `Python` code more continuously, but at the cost of editing and writing code and text in a browser window. Most of the conveniences and affordances of text editing software is lost. While `ipython` focuses on interactive computation, `knitr` focuses on bringing together scientific document formatting and computation. From the perspective of praxiography, given that both can include code written in other languages (that is, python code can be processed by `knitr`, and `R` code executed in `ipython`), the differences are not crucially important [ˆP3]. This whole book could have been written using just Python, since Python is a popular general purpose programming language, and many statistical, machine learning and data analysis libraries have been written for Python. Widely used Python modules such as NumPy, SciPy, Scikit-learn, open-cv or Pandas allow anything done in `R` to be done in Python. It is difficult to generalise about the differences between the two programming languages. I have used both, sometimes to highlight tensions between the somewhat more research-oriented `R` and the more practical applications typical of Python, and sometimes because code in one language is more easily understood than the other.

Acknowledgments

My Texas Instruments TI-97 scientific calculator conveyed in high school mathematics classes something of correlation. There I first glimpsed the idea of fitting a line to points without using just a pencil and a ruler to work out the best fit. I also would like to thank my mathematics teachers at Wollongong High for the simple pleasures of numerical optimization they first introduced to me. Newton's method for finding the minimum value of a continuous function still operates in machine learning.

Decades later, from 2007-2012, I benefited greatly from a research position in the UK Economic and Social Research Council-funded Centre for Economic and Social Aspects of Genomics at Lancaster University. Certain colleagues there, initially in the Sociomics Core Facility, participated in the inception of this book. Ruth McNally first of all, but also Paul Oldham, Maureen McNeil, Richard Tutton, and Brian Wynne were participants in many discussions concerning the transformation of life sciences around which my interest in machine learning first crystallised. Various academic staff in the Department of Applied Mathematics and Statistics at Lancaster University shepherded me through their post-graduate training courses: Brian Francis for his course of 'Data Mining,' David Lucy for his course on 'Bayesian Statistics', Thomas Jakl for his course 'Genomic Data Analysis' and TBA's course on 'Missing Data.' My colleagues in science studies at Lancaster, especially Maggie Mort, Lucy Suchman, and Claire Waterton have . I have been very fortunate to have worked with inspiring and adventurous doctoral students at Lancaster during the writing of this book. Lara Houston, Mette Kragh Furbo, Felipe Raglianti, Emils Kilis, Xaroula Charalampia, and Nina Ellis have all helped and indeed challenged me in different ways. Sjoerd Bollebakker very kindly updated many of the scientific literature searches towards the end of the book's writing.

Contents

List of Figures

List of Tables

Mitchell, Tom

machine learner!computer program as

Breiman, Leo

Chapter 1

Introduction: Into the Data

Definition: A computer program is said to **learn** from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, improves with experience E (Mitchell 1997, 2).

In the past fifteen years, the growth in algorithmic modeling applications and methodology has been rapid. It has occurred largely outside statistics in a new community—often called machine learning—that is mostly young computer scientists (Section 7). The advances, particularly over the last five years, have been startling (Breiman 2001, 200)

The key question isn't 'How much will be automated?' It's how we'll conceive of whatever *can't* be automated at a given time. (Lanier 2013, 77)

A relatively new field of scientific-engineering devices has become operational in the last three decades. The field is known by various names – machine learning, pattern recognition, knowledge discovery, data mining – and its machine learner!National Security Agency Skynet

data science!relation to machine learning

Google Search

Apple Siri

devices seem to have quickly migrated across scientific disciplines, business and commercial applications, industry, engineering, media, entertainment and government. Heavily dependent on calculation, they are found in breast cancer research, in autonomous vehicles, in insurance risk modelling, in credit transaction processing, in computer gaming, in face and handwriting recognition systems, in astronomy, advanced prosthetics, robots, ornithology, finance and surveillance (see the U.S. Government's `SkyNet` for one example of a machine learning surveillance system (Agency 2012)). Sometimes these devices are understood as *scientific models*, and sometimes they are understood as *operational algorithms* or machines. In very many scientific fields, publications mention or describe these techniques as part of their analysis of some experimental or observational data (as in the logistic regression classification models found in a huge number of biomedical papers). They anchor the field of 'data science' (Schutt and O'Neil 2013). Not so recently, they became mundane mechanisms, lying somewhere quite deeply embedded in other systems or gadgets (as in the decision tree models used in some computer game consoles to recognise gestures or the neural networks used to recognise voice commands by search engine services such as `Google Search` and `Apple Siri` (McMillan 2013)). In other settings, they operate behind the scenes as part of the everyday functioning of services ranging from player ranking in online games to border control face recognition, from credit scores to advanced full limb prosthetics. The flexibility or generality of these machine learners, their proliferation and propagation in the world, and the epistemic-operational value accruing to them by virtue of their capacity to 'learn from experience' are the concerns of this book.

The volume and geography of searches on Google Search provides some evidence of interest in particular search topics over a period of roughly a

Google!Google Trends

artificial intelligence

data mining

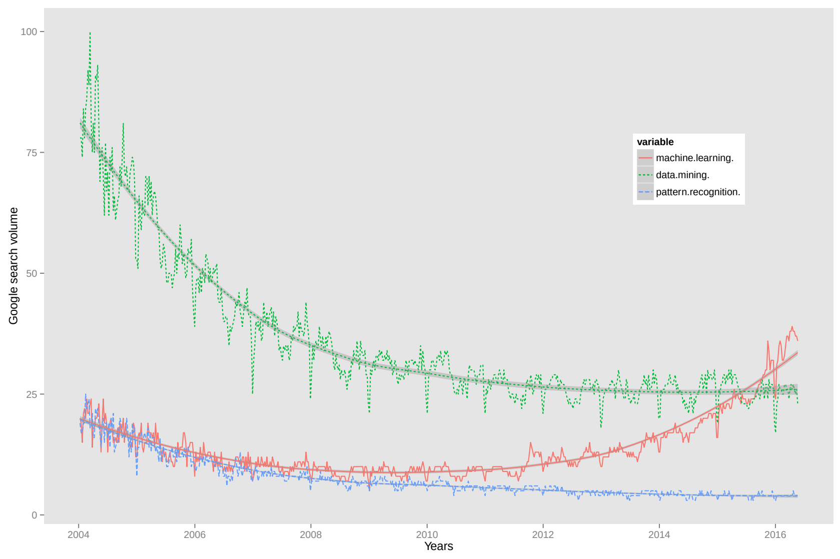


Figure 1.1: Google Trends search volume for ‘machine learning‘ and related query terms in English, globally 2004-2015

decade. If we search for terms such as on the Google Trends service , the results for the last decade or so suggest changing interest in these topics.

In Figure 1.1, two search terms that had a very high search volume in 2004 – ‘artificial intelligence’ and ‘data mining’ – slowly decline over the years before starting to increase again in the last few years. By contrast, ‘machine learning,’ ‘deep learning,’ and ‘predictive analytics’ tend to increase during that decade. Midway between `artifical intelligence` and `deep learngin`, `machine learning` appears prominently in 2004, loses volume until around 2008, and then gradually rises again so that by mid-2015 it roughly matches the long-standing interests in data-mining and artificial intelligence.[1] In the plot (Figure 1.1, the weekly variations in search volume on Google give rise to many spikes in the data. These spikes can be linked to specific invents such as significant press releases, public debates, media attention and film releases.

1. It is hard to know who is doing these searches. The data provided by Google Trends includes geography, and it would be interesting to compare the geographies of interest in the different terms over time.

Cleveland, William
local regression
k-nearest neighbors
linear regression
data!practice
kittydar
image recognition

The graphics shown in Figure 1.1 actually draw two lines for each trend. The 'raw' weekly GoogleTrends data – definitely not raw data, as it has been normalized to a percentage (Gitelman 2013) – appears in the very spiky lines, but a much smoother line shows the general trend. This smoothing line is the work of a statistical model – a local regression or loess model (Cleveland, Grosse, and Shyu 1992) developed in the late 1970s . The line depends on intensive computation and models (linear regression, k nearest neighbours,). The smoother lines make the spiky weekly search counts supplied by Google much easier to see. They construct alignments in the data by replacing the heterogeneous variations with something that unequivocally runs through time with greater regularity. The smoothed lines shade the diagram with a predictive orientation. The lineaments of machine learning already appear in such lines. How have things been arranged so that smooth lines run through many variations in this data?

```
## Traceback (most recent call last):
##   File "<string>", line 6, in <module>
## AttributeError: 'module' object has no attribute 'io'
```

What does it mean that machine learners surface in so many different places, from fMRIs to Facebook, from fisheries management to Al Queda courier network monitoring? This book is an attempt to answer that question by describing the general horizon of machine learning as a data practice in its specificity. Take the case of `kittydar`, a machine learner in the area of image recognition (see kittydar): 'Kittydar is short for kitty radar. Kittydar takes an image (canvas) and tells you the locations of all the cats in the image' (Arthur 2012) . This playful piece of code demonstrates the deployment of a machine learner in the mundane, not to say banal, domain of cat photos on the internet. Heather Arthur, who developed `kittydar` writes:

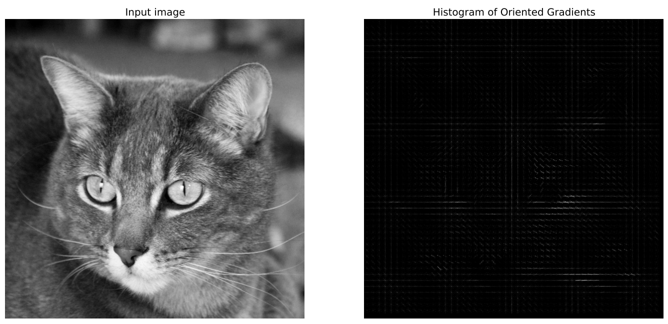


Figure 1.2: Close up of cat. The image on the left is already signal processed as JPEG format file. The image on the right is further signal processed using histogram of oriented gradients (HOG) edge detection. `kittydar` models HOG features. Photo courtesy photos-public-domain.com

> Kittydar first chops the image up into many "windows" to test for the presence of a cat head. For each window, kittydar first extracts more tractable data from the image's data. Namely, it computes the Histogram of Orient Gradients descriptor of the image, using the hog-descriptor(http://github.com/harthur/hog-descriptor) library. This data describes the directions of the edges in the image (where the image changes from light to dark and vice versa) and what strength they are. This data is a vector of numbers that is then fed into a neural network(https://github.com/harthur/brain) which gives a number from 0 to 1 on how likely the histogram data represents a cat. The neural network (the JSON of which is located in this repo) has been pre-trained with thousands of photos of cat heads and their histograms, as well as thousands of non-cats. See the repo for the node training scripts (Arthur 2012).

This toy example of machine learning data practice finds cat heads in digital photographs. Based on how it locates cats, we can begin to imagine classification

neural network!seemachine learner!neural net

support vector machine!seemachine learner!support vector machine

linear regression

machine learner!logistic regression

machine learner!neural net

machine learner!linear discriminant analysis

support vector machine

k-means clustering

\

nearest neighbours|seemachine learner!_k_ nearest neighbours

principal component analysis

machine learner!Naive Bayes

similar pattern recognition techniques in use in self-driving cars, border security systems, military robots or wherever there is something to be seen. `Kittydar` runs in Javascript in a web browser, and only detects the presence of cats that face forward. As Arthur's description suggests, the software finds cats by cutting the images into smaller windows. For each window, it measures a set of gradients running from light and dark, and then compares these measurements to the gradients of known cat images (the so-called 'training dataset'). The device associates sudden shifts from light to dark with the regular features on cats' faces. The work of classification according to the simple categories of 'cat' or 'not cat' is given either to a neural network (as discussed in chapter **??**), a typical machine learning technique and one that has recently been heavily developed by researchers at Google (Le et al. 2011), themselves working on images of cats among other things taken from Youtube videos (BBC 2012) , or to a support vector machine, a technique first developed in the 1990s by researchers working at IBM (see chapter **??**).

Machine learners range from the mundane to the esoteric, from the miniature to the infrastructural sublime of computational clouds. Like `kittydar`, they often classify, rank, cluster and estimate things. Their names accumulate and repeat in textbooks, instructional courses, website tutorials, software libraries and code listings: linear regression , logistic regression, neural networks , linear discriminant analysis , support vector machines, k-means clustering, decision trees decision tree!see{machine learner!decision tree}, *k* nearest neighbours, random forests, principal component analysis, or naive Bayes classifier to name just some of the most commonly used. Sometimes they have proper names: `RF-ACE`, `Le-Net5`, or `C4.5`. These names refer to predictive models and to computational algorithms of various ilk and provenance. Dauntingly intricate practices – normalization, regularization, cross-validation, feature engineering, feature selection, optimisation – suture datasets into shapes they can recognise. The techniques, algorithms and models are not necessarily startling new or novel. They take shape against a background of more than a century of work in mathematics, statistics, computer science as well as various scientific fields ranging from anthropology to zoology. Mathematical constructs drawn from linear algebra, differential calculus, numerical optimization and probability theory pervade practice in the field. [^0.15]

data!practices
mathematics
machine learner|textbf
positivity!of knowledge
linear discriminant analysis|seemachine learner!linear discriminant analysis

I am focusing on machine learners – a term that refers to both humans and machines throughout this book – and not algorithms, databases, infrastructures, or data visualization more generally because the data practices associated with machine learning delimit a specific *positivity* of knowing. Limiting the focus to machine learning involves some risks. On the one hand, single minded focus on machine learning or particular machine learners such as neural nets or linear discriminant analysis risks fetishizing or essentializing machine learners as machines or algorithms. To attribute primacy to the algorithm or the predictive model would be to uncritically re-iterate many contemporary performances and representations that privilege machines and marginalize their human others. On the other hand, to favour analysis of the economically, socially, and political charged contexts of machine learning would be to downplay the positivity, the relational potentials and eventfulness that might inhabit their operations. Although vital, a contextual analysis might lose sight of the transformed things, scales, processes and practices taking shape as machine learners, human and non-human, shade into each other. While `kittydar` locates cats, other machine learners disaggregate cancer sub-types, patterns of speciation in birds, identify Al-Qaeda operatives or attune the movements of a prosthetic hand to nerve activations in an amputees' residual limb. Power and knowledge, control

control
generalization
Foucault, Michel
operation, field of

and positivity mix thoroughly in machine learning. I find it difficult to separate the novel human-machine relations that have been taking shape in machine learners in their knowledge-making practices from the systems of control, classification, decision, power and discrimination they also operate. I suspect I'm not alone in that difficulty, and for that reason, I attempt in the chapters that follow to carefully describe practices, knowledges, forms and terrains specific to machine learning.

Machine learners today circulate into domains that lie far afield of the eugenic and psychology laboratories, industrial research institutes or specialized engineering settings in which they first took shape (in some cases, such as the linear regression model or principal component analysis, more than a century ago; in others such as support vector machines or random forests, in the last two decades). If they are not exactly new and have diverse genealogies, the question is: does something important happen as machine learners shift from being mathematical or engineering techniques to an everyday device that can be generalized to locate cats in digital images, the Higgs boson in particle physics experiments or fraudulent credit card transactions? Does the somewhat unruly generalization of machine learning across different epistemic, economic, institutional boundaries attest to a re-definition of knowledge, decision and control, a new operational field, as the philosopher Michel Foucault might put it, for knowledge (Foucault 1972, 106)?

All control to the machine learners?

Machine learners operate as programs. Machine learning can be viewed as a change in how programs or the code that controls computer operations are written and operate. From a control perspective, machine learners continue the 'control revolution' that arguably has, since the late nineteenth century, programmatically re-configured production, distribution, consumption, and bureaucracy (Beniger 1986). With the growth of communication networks, the late 20th century suffered a new crisis of control, commensurate with the mid-19th century crisis described by James Beniger (Beniger 1986). Almost all accounts of the operation power of machine learning emphasise its power to automate the control of processes – border flows, credit fraud, spam email, financial market prices, gene expression, targetted online adverts – whose unruly or transient multiplicity otherwise evades or confuses us.

control!control revolution

facial recognition

Amoore, Louise

Munster, Anna

\

If a new programmatic field of knowledge-control takes shape around machine learning, how would we recognize it? In a study of border control systems, which often use machine learners to do profiling and facial recognition , Louise Amoore advocates attention to calculation and algorithms:

> Surely this must be a primary task for critical enquiry – to uncover and probe the moments that come together in the making of a calculation that will automate all future decisions. To be clear, I am not proposing some form of humanist project of proper ethical judgement, but rather calling for attention to be paid to the specific temporalities and norms of algorithmic techniques that rule out, render invisible, other potential futures (Amoore 2011).

As Amoore writes, some potential futures are being 'ruled out' as these devices are put to work. Anna Munster puts the challenge more bluntly: 'prediction takes down potential' (Munster 2013). I find much to agree with here. Machine learning is a convoluted but nevertheless concrete and historically specific form of calculation calculation|see{mathematics!calculation} (as we will see in exploring algebraic operations in chapter **??**, in decision

infrastructure

handwriting recognition|seealsodigit recognition

algorithm!primacy

finding and optimising certain mathematical functions in chapter 3 or in characterising and shaping probability distributions in chapter **??**). It tends to algorithmically mediate future-oriented decisions (although all too often, very near-future decisions). It automates, and in many cases, specifically aims to automate that which hitherto appeared impossible to automate. And this automation, with all the investment it attracts (in the form of professional lives, in the form of infrastructures , in research funding, in reorganisation of corporate and government processes, etc.) does rule out some and reinforce other futures. As for consequences, we need only consider some of the many forms of work that have already been affected by or soon could be affected by machine learning. Postal service clerks no longer sort the mail because neural net-based handwriting recognition reads addresses on envelopes . Locomotives, cars and trucks are already driven by machine learners, and soon driving may not be same occupational cultural it was. Hundreds of occupational categories have to some degree or other machine learners in their near future.[2]

Given the consequential weight of such algorithms and calculations, how do we uncover the moments that come together in them? We might see algorithms as making calculation automatic. In various scholarly and political debates around changes business, media, education, health, government or science, quasi-omnipotent agency has been imputed to algorithms [Pasquinelli (2014);Neyland (2014);Totaro and Ninno (2014);M. Smith (2013); Beer and Burrows (2013);Fuller and Goffey (2012);Wilf (2013); Barocas, Hood, and Ziewitz (2013); Gillespie (2014); (Cheney-Lippold 2011; A. R. Galloway 2004) or sometimes just 'the algorithm.' The power of algorithms in the social science and humanities literature is understood in different ways, but there is general agreement that algorithms are powerful, or at least,

2. Carl Benedikt Frey and Michael Osborne model the chances of occupational change for 700 occupations using, aptly enough, the machine learning technique of Gaussian Processes (Frey and Osborne 2013).

can bear down heavily on people's lives and conduct, re-configuring, for instance, culture as algorithmic (Hallinan and Striphas 2014). In what does this power and increasing prestige consist? What imbues algorithms with power? Is there some general or 'breakout' feature such as recursivity) or abstraction that configures contemporary rationality algorithmically?

algorithm!recursivity
abstraction!in algorithms
Pasquinelli, Paolo
abstraction!algorithm as

Much of the critical literature on algorithms identify mathematical or predictive aspects as the source of their power. For instance, in his discussion of the 'metadata society,' Paolo Pasquinelli proposes that

> a progressive political agenda for the present is about moving at the same level of abstraction as the algorithm in order to make the patterns of new social compositions and subjectivities emerge. We have to produce new revolutionary institutions out of data and algorithms. If the abnormal returns into politics as a mathematical object, it will have to find its strategy of resistance and organisation, in the upcoming century, in a mathematical way (Pasquinelli 2015).

'Moving at the same level of abstraction as the algorithm' offers some purchase as a formulation, but I find 'the' algorithm, 'the level of abstraction' and 'a mathematical way' all troublesome as ways of qualifying machine learners. Which algorithm, what kind of abstraction and which mathematical way should we focus on? From the standpoint of diverse machine learners and the many different person-thing-object-relations they encompass, there is no single a-historical level of abstraction, but something more like a torque and flux of different moments of abstraction at work in generalizing, classifying, circulating and stratifying in the midst of transient and plural multiplicities. [^0.12] automation!limits of

machine learner!diagrammatic composition of

mathematics!as stabilisation of practice

The archaeology of a data practice

The coming together of moments in machine learning also have a play in them that notions of automation and algorithm do not fully accommodate. Terms such as automation and algorithm over-prune machine learners as forms of data practice. As I will suggest, algorithms, calculations, techniques and data cohere as machine learners in specific ways. This makes them harder to see as devices or indeed in context. While they can be contextualised in industry, science or government, they themselves contextualise data, decisions, classifications and rankings. Viewed as an ordering practice, we might analyse how machine learners diagonally weave, layer and leverage techniques, calculations and algorithms in infrastructures, institutions and everyday lives. With the certain variations in focus (for instance, paying close attention to the substitution and connection of various elements; see chapter 2), could we see some stable forms – and these often achieve mathematical formalization as they become stable – amidst the maelstrom of platforms, devices, skills, claims and advocates flowing around them? We might see change that occurs more slowly than what flows around them. Understood as a data practice, we can begin to see the emergence of regularities and forms of order that allow higher levels of abstraction – the algorithm, the data, the mathematical, indeed, abstraction itself – to cohere.

If abstractions, mathematics or the algorithm loom large in classification practices today, how do we re-establish their connection to the workings of power without pre-emptively ascribing potency to the mathematical way, to the algorithm or abstraction? In the chapters that follow, I map machine learning data practices in greater empirical and conceptual depth, and develop some terms to describe them less generically (common vector space in Chapter **??**, partial observer trajectory in Chapter 3, decision surface in

Chapter **??**, inverse probabilization in Chapter **??**), but all of the facets I describe cluster around the problem of understanding how an almost banal, not esoteric, operation is repeated, borrowed, copied and diffused in so many settings.

Couldry, Nick
categories
power

If we understand machine learning as a specific data practice, then the forms of order that result from it also become more tangible. Nearly all machine learners can classify things. They are often simply called 'classifiers.' `Kittydar` classifies images as cats, but categorisation and classification applies much more generally.[3] Many machine learners predict categories, levels, rankings, and values (for instance, prices or risks – see Chapter **??**). In his account of media power, Nick Couldry highlights the importance of categories and categorisation:

> *Category* is a key mechanism whereby certain types of ordered (often 'ritualized') practice produce power by enacting and embodying categories that serve to mark and divide up the world in certain ways. Without *some* ordering feature of practice, such as 'categories', it is difficult to connect the multiplicity of practice to the workings of power, whether in the media or in any other sphere. By understanding the work of categories, we get a crucial insight into why the social world, in spite of its massive complexity still appears to us as a *common* world (Couldry 2012, 62) ,

The orderings of practice, effected through categories, undergo a great deal of intensification via machine learning. While Couldry does not in this

3. John Cheney-Lippold offers a quite general overview of categorization work. He writes: 'algorithm ultimately exercises control over us by harnessing these forces through the creation of relationships between real-world surveillance data and machines capable of making statistically relevant inferences about what that data can mean' (Cheney-Lippold 2011, 178). . Much of my discussion here seeks to explore the space of 'statistical inference of what that data can mean.'

data!training

machine learning!regularization

context discuss data mining, machine learning or predictive analytics, his analysis of categorisation and its contribution to a massively complex but common social world points directly to the order of practice of practice.

Machine learning might be seen as an ordering practice that marks up and divides the world. The operation of classification or prediction usually depends on a set of training data whose categories are already known or have been assigned by someone (expert or not). These categories are sometimes simply an existing set of classifications derived from institutionalised or accepted practice. Machine learners also generate new categorical workings or mechanisms of category creation. Sometimes, new sets of categories have been invented or found for a particular purpose. The person who finds themselves paying a different price for a holiday by virtue of some unknown combination of factors including age, credit score, home address, previous travel, or educational qualifications experiences something of the diagrammatic movements.

From this perspective, the mathematical abstractions, whether they come from calculus, linear algebra, statistics, or topology, maximise regularities. The power of machine learning to find patterns, to classify or predict regugularizes ordering practice. . The different facets of ordering practice I discuss arrange data in certain ways and not others, traverse data along certain lines, curves and surfaces, pay close attention to variations and generalization. These practices precede, prepare, shape and limit the algorithms, the abstractions and their predictions or classifications.

Massive asymmetries in a common world

If we see a massive regularization of order occurring in machine learning, what is at stake in trying to think through those practices? They display moments of formalisation (especially mathematical and statistical), circulation (pedagogically and operationally), generalization (encompassing many genera and generic forms of practice) and stratification (the socially, epistemically, economically and sometimes politically or ontologically loaded re-iterative enactment of categories). If that ordering becomes more tangibly thinkable, would it change how would we relate to what we see, feel, sense, hear or think in the face of a contemporary webpage such as Amazon's that uses Association Rule Mining , an app, a passport control point that matches faces of arriving passengers with images in a database, a computer game, or a genetic test (all settings in which machine learning is likely to be operating)?

Association Rule Mining|seemachine learner!\textitapriori algorithm

machine learning!as ordering practice

Hammerbacher, Jeff

advertising, online

The ordering practices of machine learning display some rather stunning uniformities. Some expert practitioners complain of this uniformity. Jeff Hammerbacher , previously chief research scientist at Facebook, co-founder of a successful data analytics company called Cloudera, and currently working also on cancer research at Mount Sinai hospital, complained about the spread of machine learning in 2011: 'the of my generation are thinking about how to make people click ads' (Vance 2011) . Leaving aside debates about the ranking of 'best minds' (see chapter **??**), Hammerbacher was referring to the flourishing use of predictive analytics techniques in online platforms such as Twitter, Google and Facebook, and on websites more generally, whether they be websites that sell things or advertising space. The mathematical skills of many PhDs from MIT, Stanford or Cambridge were wrangling data in the interests of micro-targeted advertising. As Hammerbacher observers, they were 'thinking about how to make people click ads,' and this 'thinking' mainly took and does take the form of building predictive models that tailored the ads shown on websites to clusters of individual preferences and desires. In thinking about individual people,

abstraction!equations as
digital humanities
Jockers, Matthew
topic model
Mohr, John
data!latent variables in

and indeed, in seeking to figure an individual amidst very large numbers of people (often numbering millions and sometimes hundreds of millions), the 'best minds' were also constructing and operating with the very forms of abstraction we see in Equation **??**.

Hammerbacher's unhappiness with ad click prediction resonates in critical responses to machine learning as used in other settings. Some versions of the digital humanities make extensive use of machine learning. To cite one example, *Macroanalysis: Digital Methods and Literary History*, Matthew Jockers describes he or we might relate to one currently popular machine learning or statistical modelling technique, the topic model (itself the topic of discussion in Chapter **??**; see also (Mohr and Bogdanov 2013)):

> If the statistics are rather too complex to summarize here, I think it is fair to skip the mathematics and focus on the end results. We needn't know how long and hard Joyce sweated over *Ulysses* to appreciate his genius, and a clear understanding of the LDA machine is not required in order to see the beauty of the result. (Jockers 2013, 124)

The widely used topic models or Latent Dirichlet Allocation models provide a litmus test of how relations to machine learning is taking shape in the digital humanities. On the one hand, these models promise to make sense of large corpus of documents in terms of underlying themes or 'topics.' Latent topics and latent variables are of much more general interest in predictive modelling and classification (as we will, large document collections have long attracted the interest of machine learners). On the other hand, Jockers signals the difficulties of relating to machine learning when he suggests that 'it is fair to skip the mathematics' for the sake of 'the beauty of the result'. While one part of the humanities and critical social research exhorts closer attention to the mathematical, another averts its gaze in face of their complexity.

Galloway, Alex

decision tree

The use of machine learning in digital humanities has not always been received enthusiastically. In a special issue of the journal *Differences: A Journal of Feminist Cultural Studies* focusing on digital humanities, Alex Galloway makes two observations about the circulation and generalization of these methods in humanities scholarship:

> When using quantitative methodologies in the academy (spidering, sampling, surveying, parsing, and processing), one must compete broadly with the sorts of media enterprises at work in the contemporary technology sector. A cultural worker who deploys such methods is little more than a lesser Amazon or a lesser Equifax.110 (A. Galloway 2014, 110)

Galloway is critical of the asymmetry between humanities scholars and media enterprises and credit score agencies. The 'quantitative methodologies' that he refers to as spidering, sampling, processing and so forth are more or less all epitomised in machine learning techniques (for instance, the Association Rule Mining techniques used by Amazon to recommend purchases, or perhaps the decision tree techniques used by the credit-rating systems at Equifax and FICO (Fico 2015)). Galloway's argument is that the infrastructural scale of these enterprises along with the sometime very large technical workforces they employ to continually develop new predictive techniques dwarfs any gain in efficacy that might accrue to humanities research in its recourse to such methods.

Even if 'cultural workers' do manage to learn to machine learn, and become adept at re-purposing the techniques in the interests of something other than selling things or generating credit scores, what is to be gained by doing so?

Galloway, Alex|)

Galloway suggests that they might actually reinforce power asymmetries and exacerbate the ethical and political challenges posed by machine learning:

> But beyond the challenge of unequal talent and resources is the question of critical efficacy. Is it appropriate to deploy positivistic techniques against those self-same positivistic techniques? In a former time, such criticism would not have been valid or even necessary. Marx was writing against a system that laid no specific claims to the apparatus of knowledge production itself—even if it was fueled by a persistent and pernicious form of ideological misrecognition. Yet, today the state of affairs is entirely reversed. The new spirit of capitalism is found in brainwork, self-measurement and self-fashioning, perpetual critique and innovation, data creation and extraction. In short, doing capitalist work and doing intellectual work—of any variety, bourgeois or progressive—are more aligned today than they have ever been (A. Galloway 2014, 110).

This perhaps is a more serious charge. The 'techniques' of machine learning are positivist (and hence implicitly at odds with critical thought?), and moreover complicit – 'aligned' – with capitalist work. Again, there is something that feels right in the naming of the predicament – intellectual work of the kind associated with machine learning – is definitely at the centre of many governmental, media, business and scientific fields of operation. Increasingly, they anchor the operations of these fields.

What cannot be automated?

Like Galloway, I'm wary of certain deployments of machine learning, particularly the platform-based deployments and their associated sociality

(Gillespie 2010; Van Dijck 2012). In certain case, they do seem to be 'laying claim to the apparatus of knowledge production.' Yet even amidst the trashy ephemerality of targeted online advertising or the more elevated analytics of literary history, the transformations in knowledge and knowing do not simply align intellectual work and capitalist work. Machine learning as a data practice is not simply automating existing economic relations, even if that reproduction heavily steers its normal practice. While Hammerbacher and Galloway are understandably somewhat dismissive of the existential gratifications and critical efficacy of building targeted advertising systems or document classifiers, the 'deployment' of machine learning is not a finished process, but very much in train, constantly subject to revision, re-configuration and alteration. Could machine learners become engines of difference? Where in the algorithms, calculations, abstractions and regularizing practices of machine learning would differences be re-draw?

phronesis!predictive

Machine learning in journalism, in specific scientific fields, in the humanities, in social sciences, in art, media, government or civil society sometimes overflows the platform-based deployments and their trenchantly positivist usages. A fairly explicit awareness of the operation of machine-learning driven processes is taking shape in some quarters. And this awareness couples critical and practical responses in a situationally aware calculative knowledge-practice, a form of predictive phronesis (Aristotle 1981). For instance, the campaign to re-elect Barack Obama as U.S. President in 2011-12 relied heavily on micro-targetting of voters in the leadup to the election polls (Issenberg 2012; Mackenzie et al. 2016). In response to the data analytics-driven election campaign run by the US Democrats in support of the 2012 re-election of President Barack Obama, data journalists at the non-profit news organisation *ProPublica* reverse engineered the machine learning models that allowed the Obama re-election team to target natural language processing

Bogost, Ian

Netflix

Google!Google Flu

individual votes with campaign messages (Larsen 2012). They built their own machine learning model - the 'Message Machine' - using emails sent in by readers and supporters to identify the workings of the Obama campaign team's micro-targetting models. While the algorithmic complexity and data infrastructures used in the Message Machine hardly match those at the disposal of the Obama team, it combines natural language processing (NLP) techniques such as measures of document similarity and machine learning models such as decision trees to disaggregate and map the micro-targetting processes . This kind of reverse engineering work can be found in other quarters. In response to the personalised recommendations generated by streaming media service Netflix, journalists at *The Atlantic* working with Ian Bogost, a media theorist and programmer , reverse engineered the algorithmic production of around 80,000 micro-genres of cinema used by Netflix (Madrigal 2014) . While Netflix's system to categorise films relies on much manual classification and tagging with meta-data, the inordinate number of categories they use is typical of the classificatory regimes that are developing in machine learning-based settings. Certainly, high-profile claims for the power of predictive models to pre-emptively forecast events has come into question. The epidemic predictions of the Google Flu system, a predictive model based on the geography of search engine queries, were wrong on several occasions, mainly because people's online search behaviour changed as a result of coverage (Lazer et al. 2009; Butler 2013) .

While these cases may be exceptional achievements, and indeed highlight the suffocating weight of the ad-tech application of machine learning, the proliferation of scientific usage suggests that the generalization of machine learning cannot be reduced to personalized advertising. Despite their many operational deployments, the coming together of algorithm, calculation and technique in a form of data practice is not fully coherent or complete. In order to qualify or specify how machine learners exist in their generality, we would need to specify their operations at a level of abstraction that neither attributes a mathematical essence to them nor frames them as producers of relative surplus value. Finding ways of accommodating their diversity, loose couplings and mutability would mean grasping their operational power and their capacity to create new forms of difference.[4]

Title	Year
Regional climate change in Portugal: Precipitation variability associated with large-scale atmospheric circulation	1998
Integrating climate forecasts and natural gas supply information into a natural gas purchasing decision	2000
Potential effects of climate change on birds of the Northeast	2008
Aspen, climate, and sudden decline in western USA	2009
Reducing Peak Electricity Demand in Building Climate Control using Real-Time Pricing and Model Predictive Control	2010
21st century climate change threatens mountain flora unequally across Europe	2011
Linking climate, gross primary productivity, and site index across forests of the western United States	2011
SVM, ANFIS, regression and climate based models for reference evapotranspiration modeling using limited climatic data in a semi-arid highland environment	2012
Can Top-of-Atmosphere Radiation Measurements Constrain Climate Predictions? Part II: Climate Sensitivity	2013
Tree-species range shifts in a changing climate: detecting, modeling, assisting	2013

Table 1.1: A small sample of titles of scientific articles that use machine learning in relation to ”climate”

Different abstractions in machine learning?

Table 1.1 presents a small sample of scientific literature at the intersection of ‘climate’ and machine learning. This sample, while no doubt dwarfed by the

4. Certain strands of social and cultural theory have taken a strong interest in algorithmic processes as operational forms of power. For instance, the sociologist Scott Lash distinguishes the operational rules found in algorithms from the regulative and constitutive rules studied by many social scientists:

> in a society of pervasive media and ubiquitous coding, at stake is a third type of rule, algorithmic, generative rules. ‘Generative’ rules are, as it were, virtuals that generate a whole variety of actuals. They are compressed and hidden and we do not encounter them in the way that we encounter constitutive and regulative rules. Yet this third type of generative rules is more and more pervasive in our social and cultural life of the post-hegemonic order. They do not merely open up opportunity for invention, however. They are also pathways through which capitalist power works, in, for example, biotechnology companies and software giants more generally (Lash 2007, 71).

The term ‘generative’ is somewhat resonant in the field of machine learning as generative models, models that treat modelling as a problem of specifying the operations or dynamics that could have given rise to the observed data, are extremely important. If we consider only Andrew Ng’s CS229 machine learning lectures on Youtube (*Lecture 1 | Machine Learning (Stanford)* 2008) (lectures I discuss in chapter 2 and draw on throughout this book), we can see that they introduce generative models in Lecture 5 and 6. Although this seems to be only a small part of the 18 lectures given in the course, later lectures on the expectation maximisation algorithm (12-13), and then on unsupervised learning techniques such as factor analysis and principal component analysis, independent component analysis, flood of publications on recommendation systems, targeted advertising or handwriting recognition, is typical of the epistemic atmosphere associated with machine learners. (I return to this topic in Chapter **??** in discussing how the leveraging of scientific data via predictive models and classifiers deeply affects the fabric and composition of objects of scientific knowledge.) But the longevity and plurality of experiments, variants, alternative techniques, implementations and understandings associated with machine learning makes it difficult to immediately reduce them to capitalist captures of knowledge production.

scientific publications

Figure 1.3 derives from counts of scientific publications that mention particular machine learners in their title, their abstract or keywords. The curves, which are probability density plots, suggest a distribution of statements and operations over time for different techniques. Both the duration and the ebbs and flows of work on specific techniques, platforms, knowledges and power relations is still largely occluded. Like the Google Trends results, the lines shown in Figure 1.3 have been normalised in order to adjust for an overall increase in the volume of scientific publications over the last five decades. Unlike the Google Trends search patterns, the scientific literature displays a much more granular and differentiated texture in which different techniques, different terms over the last half century diverge widely from each other. A quick glance at science shows less homogeneity than

are also effectively exploring generative models. A similar distribution of topics can be found in *Elements of Statistical Machine Learning*(Hastie, Tibshirani, and Friedman 2009). Generative models, while perhaps slightly less common in practice than discriminative models, nevertheless capture the sense that algorithms are not just implementations of rules for filtering, sorting, or deciding, but carry within them ontological commitments that might actually challenge social theory in interesting ways. In contrast to Lash, I would suggest that the generativity of these algorithms needs to be differentiated from the algorithmic processes that implement rules more generally. Moving into the data via a generative probabilistic model is very different to moving into the data through say a database query. The models, whether generative or discriminative (models such as decision tree, logistic regression or even neural networks that are more limited in their probabilistic underpinnings), are more like meta-algorithms that reorganize other algorithmic processes on varying scales.

abstraction!accounts of

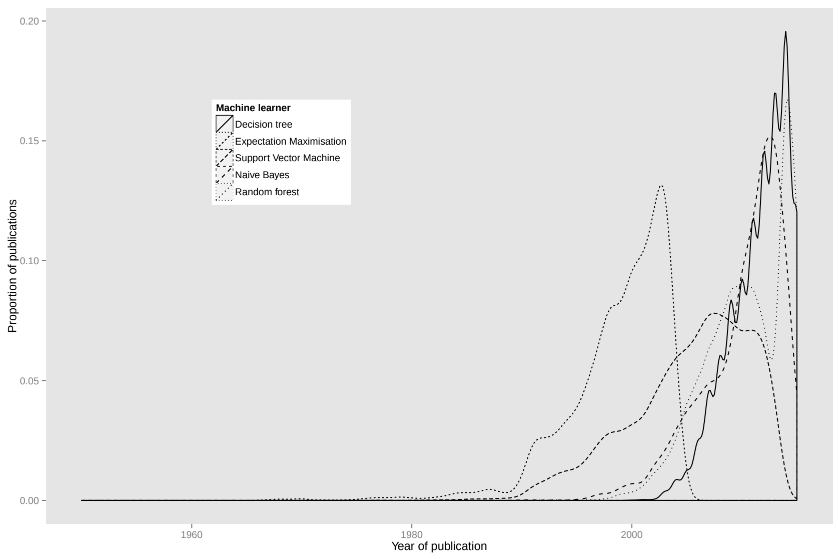


Figure 1.3: Machine learners in scientific literature. The lines in the graph suggest something of the changing fortunes of machine learners over time. The publication data comes from Thomson Reuter's *Web of Science*. Separate searches were run for each machine learner. In these plots, as in the GoogleTrends data, the actual counts of publications have been normalised. In contrast to the GoogleTrend plots, these plots do not show the relative counts of the publications, only their distribution in time.

Hammerbacher and perhaps Galloway see.

Given some degree of pluralism in machine learning as a data practice, what level of abstraction usefully specifies it? If the algorithm and the mathematical constrict it too much, what other level of abstraction could we turn to? Abstraction stands out as one of the most richly developed theoretical resources in the social sciences and humanities. Accounts of abstraction – Karl Marx's real abstraction, Gilles Deleuze and Félix Guattari's abstract machines, Henri Bergson's lived abstraction, Alfred North Whitehead's fortunate abstraction, or Isabelle Stengers' experimental abstractions – are not lacking.[5] Although the theories of abstraction I have just listed differ in many ways, they all, without exception, array themselves in opposition to any limitation of abstraction to mathematical or modern scientific knowl-

5. (McCormack 2012) reviews some of the large literature on abstraction. The differences between different accounts of abstraction will run through many of the following chapters.

edge in particular. All of them, by often convoluted conceptual paths, seek to retrieve from abstraction something conducive to change, differences and contestation.

abstraction!operational practice of
abstraction!lived

Any theory of abstraction faces a real test when confronted by operational practices of abstraction such as machine learning. Can a theory of abstraction affirm an operational abstraction without omitting its technical practice? Is machine learning an abstraction we can live with, a lived abstraction as Brian Massumi would call it (Massumi 2013)? I find the question of whether machine learning is a liveable abstraction too hard to answer conclusively. Certainly, accounts of abstraction – abstract machine, real abstraction for instance – help make sense of important movements in machine learning. But without grounding these abstractions in the practice of machine learning, it is very hard to sense how it abstracts. Without tracing how number, chance, classification and event come together in it, the texture of its abstraction, and hence any possible sense of its liveable relationality, remains unfelt and unthought.

The diagram

Much of this book attempts to identify good levels of abstraction for the data practices of machine learning. The diagram – a form of drawing that smooths away many of the frictions and variations in drawing – anchors much of my account (see chapter 2 for a fuller framing). Diagrams practically abstract. They retain a connection to doing things, such as learning, that other accounts of abstraction sometimes struggle with. Perceptually and technically, they span and indeed criss-cross between human and machine machine learners. They exhibit compositional characteristics of substitution, variation and superimposition, as well as a play or movement amongst their machine learner!SkyNet

elements. By virtue of its diagrammatic composition, machine learning might bring something new into the worlds it traverses. If machine learners reorganise data, calculation, classification, decision, control and prediction, that might have some precedent. But precedent or novelty would be quite hard to grasp without being able to trace its diagrammatic composition. Similarly, in order to understand the operational forms of power associated with machine learning, the connections connecting data structures, infrastructures, processors, databases and lives might be traceable in diagonal lines, in overlays, and indeed in the densely operational indexes of mathematical formalisms. For instance, how many cat photos are on the internet? This empirical question might be answerable only through machine learning. `kittydar` would need 300,000 cores on Google Compute Engine for several hours. Similarly, NSA's `Skynet` purports to identify Al-Qaeda couriers in Pakistan, but seems to also detect well-known Al-Jazeera journalists (Agency 2012). Even the existence of such egregious errors or any other claim to know predicated by `Skynet` could be understood as a diagrammatic practice.

Above all, diagrams can be implemented in multiple ways. Using implementations, graphical and mathematical forms, books and the heavy accumulation of scientific publications from many disciplines (for instance, as seen in figure 1.3), the book follows six main operations diagrammatically: vectorisation, optimisation, probabilisation, pattern recognition, regularization and propagation. These generic operations compose a diagram of machine learning spanning hardware and software architectures, organizations of data and datasets, practices of designing and testing models, intersections between scientific and engineering disciplines, ongoing historical transformations of notions of pattern, class, rank and order, professional and popular pedagogies, and their associated subject positions. With varying degrees of formalization and consistency, these operations seek to offer an alternative account of machine learning, an account in which some feeling of agency and of affective movement can take route. Where do the operations come from? They as technical processes and practices, sometimes at a quite low level (for instance, vectorisation) and other times widely distributed (for instance, in pedagogy). They become figures when drawn on a diagram. As an abstraction, the diagram of the operational power of machine learning does not map the footprint of a strategic monolith, but highlights the local relations of force that feed into the generalization and plurality of the field in both its monumental and peripheral variations. These local orderings, distributions, rankings and estimation can support hegemonic platforms, but they may also concatenate in different vectors of movement.

Domingos, Pedro
Deleuze, Gilles
diagram

Chapter 2

Diagramming machine learning

> Machine learning is not magic; it cannot get something from nothing. What it does is get more from less. Programming, like all engineering, is a lot of work: we have to build everything from scratch. Learning is more like farming, which lets nature do most of the work. Farmers combine seeds with nutrients to grow crops. Learners combine knowledge with data to grow programs. (Domingos 2012, 81)

> Diagram: 'a functioning, abstracted from any obstacle … or friction' (Deleuze 1988, 34)

Many different tendencies shape data practice today, but our sense of the potentials of data and calculation is closely linked to Pedro Domingos' contrast between 'a lot of [building] work' and the 'farming' done by machine learners to 'grow programs.' Although it sounds a bit banal, there is quite a lot at stake and many complications in characterising this growing work programmability!problem of

abstraction|(

Python|seealsoprogramming language

R|seealsoprogramming language!R

of growing, which Domingos presents as a shift in programming practice. In Domingo's talk of 'learners,' we might find it hard to decide what the end point is: a program or learning something? These tensions between programming and something else ('growing') are worth pursuing. They help make sense of an immense Magellanic constellation of documents, software, publications, blog pages, books, spreadsheets, databases, data centre architectures, whiteboard and blackboard diagrams, and an inordinate amount of talk and visual media orbiting around machine learning.

The changes in data practice associated with machine learning work on multiple levels, ranging from infrastructures and chip architectures through to mathematical functions *and* categorisations with significant social and economic implications. Abstracting as a practice takes many forms and is distributed widely, making it difficult to identify a single level of abstaction. In this chapter, therefore, I attend to the methodological problem of identifying and mapping the different abstractive practices intersecting in machine learning. These practices can be traced through code written to do machine learning, through pedagogical expositions of machine learning focused on both mathematical operations and graphical diagrams, and the forests of scientific or technical research publications, ranging from textbooks to research articles, that vary, explore, experiment and generalize machine learners. Both the coding practices and the pedagogical expositions have shifted substantially over the last decade or so due to the growth in open source programming languages and as a part of the broader and well-known expansion of digital media cultures (Couldry 2012). The fact that scientists, software developers and data analysts use programming languages such as `Python` and `R` just as much as commercial statistical and data software packages such as Matlab, SAS or SPSS is perhaps symptomatic of shifts in calculative culture (Muenchen 2014), but almost definitely crucial to the generalization of machine learning. Scientific research, which has long relied on counting and calculation, increasingly organises and processes data more comprehensively because workflows, datasets, algorithms and databases can be quickly and flexibly formatted and manipulated in code. (Scientific computing languages such as FORTRAN – 'Formula Translator' – have long underpinned scientific research and engineering applications in various fields (Campbell-Kelly 2003, 34-35).) Several decades of concentrated research and intensive development of statistical machine learners in science, government, industry and commerce have gradually developed fairly regular operations, patterns and functions for reshaping data and constructing models that classify and predict events and associations between things, people, processes, etc. When Domingos speaks of 'growing' programs, he might well be referring to the accumulation of certain well-worn data practices that laminate layers of abstraction. In some ways, the different modes of writing code are precisely the faceted levels of abstraction which we might need to access and traverse in order to know and come to grips empirically with contemporary compositions of power and subjectivity.

programming language!FORTRAN
abstraction|)
Ng, Andrew

A diagram of the elements of machine learning

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. Significantly, I also participated in the online machine learning courses and some machine learning competitions.[^1.4] These are all typical activities for people learning to do machine learning. Importantly, these courses, books, competitions and programming languages are widely viewed and used. Andrew Ng's machine learning course on Coursera (of which he is co-founder, along with Daphne

Koller, Daphne

\textitElements of Statistical Learning

Hastie, Jeff

Tibshirani, Rob

Friedman, Jerome

Koller, another machine learning scientist from Stanford) , has a typical enrolment of 100,000. Youtube videos of his Stanford University lectures have cumulative viewing figures of around 500,000 (*Lecture 1 | Machine Learning (Stanford)* 2008). Amidst this avalanching learning of machine learning, a single highly cited and compendious textbook, *Elements of Statistical Learning: Data Mining, Inference, and Prediction* (Hastie, Tibshirani, and Friedman 2009), currently in its second edition, can be seen from almost any point of the terrain.[1] At least for narrative purposes, I regard this book as a major practical assemblage, a 'mechanism of statements and visibilities' (Deleuze 1988, 51), a display of relations between forces and some unbound points of change. The authors of the book, Jeff Hastie, Rob Tibshirani and Jerome Friedman are statisticians working at Stanford and Columbia University. (Statisticians and computer scientists from Stanford University loom large in machine learning only since 2000; before that, the field remains virtually the sole province of computer scientists.)

In terms of scientific publications, *Elements of Statistical Learning* is a quasar. It is a massive textual object, densely radiant with diagrams, 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: equations = 968, figures = 291; tables = 34; and algorithms = 94, giving a total of 1387 operational devices threaded through the book). Around 968 equations rivet the text into mathematical abstractions of varying sophistication, and construct an semiotic machinery of considerable sophistication and connectivity. Importantly, on each page of the book we are seeing, reading,

1. 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.

puzzling over and perhaps learning from the products of code execution. The 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.

hyperobject

Ripley, Brian

Mitchell, Tom

Flach, Peter

In the range of references, it's 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, machine learning betrays some hyperobject-like tendencies (Morton 2013). Like the online machine learning courses and programming books I discuss below, *Elements of Statistical Learning* combines statistical science with various algorithms to 'learn from data' (Hastie, Tibshirani, and Friedman 2009, 1). 768 tersely written and quite mathematical pages range across various kinds of problems (identifying spam email, predicting risk of heart disease, recognising handwritten digits, etc.), and various machine learning techniques, methods and algorithms (linear regression, k-nearest neighbours, neural networks, support vector machines, the Google Page Rank algorithm, etc.). This is not the only juggernaut machine learning text. I could just have well cleaved to Alpaydin's *Introduction to Machine Learning* (Alpaydin 2010) (a more computer science-base account), Christopher Bishop's heavily mathematical *Pattern recognition and machine learning* (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). 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.

How does such a book help us access levels of abstraction and their attendant tensioning in practice? While it certainly does not comprehend everything taking place in and around machine learning, it operates as a kind of diagram of several *elementary* tendencies or traits. It has a heterogeneous texture in terms of the examples, formalisms, disciplines and domains it covers. It's readership as we will see is widespread. It starkly renders the problems of making sense of mathematical operations, diagrams and transformations carried on through calculation, simulation, deduction or analysis. It practically intersects with coding practices, particularly in the form of the `R` code it heavily but somewhat latently relies on. Such a panoply of materials and settings suggests a multi-faceted layering of abstractive practice. My primary concern in this chapter, therefore, will be to outline the different facets of machine learning that a book such as *Elements of Statistical Learning* brings to light.

Who reads the book? 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, Google Scholar reports over 20,00 citations of the book http://scholar.google.com/scholar?hl=en&q=elements+of+statistical+machine+learning, October 2014), a huge citation count by any standards. (Michel Foucault's *The History of Sexuality: an Introduction*, one of the most highly cited book in the humanities, receives about the same number of citations.) The citations that (Hastie, Tibshirani, and Friedman 2009) as well as its previous edition (Hastie, Tibshirani, and Friedman 2001) receive do not arrive principally from machine learning researchers. They come from a wide variety of fields. 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 results. These publications sprawl across well 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 2.1 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, the molecular biology, control and automation, operation research, business and public health soon appear, suggesting something of the propagating energy of machine learning.

So we know that *Elements of Statistical Learning* passes into many fields. But what do people read in the book? In general, the thousands of citations of the book themselves comprise a diachronic diagram of readings of the book, and their relative concentrations and sparsities suggest there may be specific sites of engagement in the techniques, approaches and machines that the book describes. Of the around 760 pages in (Hastie, Tibshirani, and Friedman 2009) and (Hastie, Tibshirani, and Friedman 2001), around 77

Non-negative matrix factorization

Citations	Field
4781	Computer Science
2935	Engineering
2146	Mathematics
937	Mathematical & Computational Biology
862	Biochemistry & Molecular Biology
618	Environmental Sciences & Ecology
501	Automation & Control Systems
491	Neurosciences & Neurology
474	Biotechnology & Applied Microbiology
455	Chemistry
450	Imaging Science & Photographic Technology
442	Science & Technology - Other Topics
423	Operations Research & Management Science
385	Radiology, Nuclear Medicine & Medical Imaging
358	Business & Economics
310	Genetics & Heredity
288	Physics
272	Remote Sensing
244	Public, Environmental & Occupational Health
233	Geology

Table 2.1: Subject categories of academic research literature citing *Elements of Statistical Learning* 2001-2015. These subject categories derive from Thomson-Reuter *Web of Science*.

distinct pages are referenced in the citing literature. As figure **??** 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 533. 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 (Lee and Seung 1999)) It is particularly powerful because 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 ‘kittydar‘

face recognition

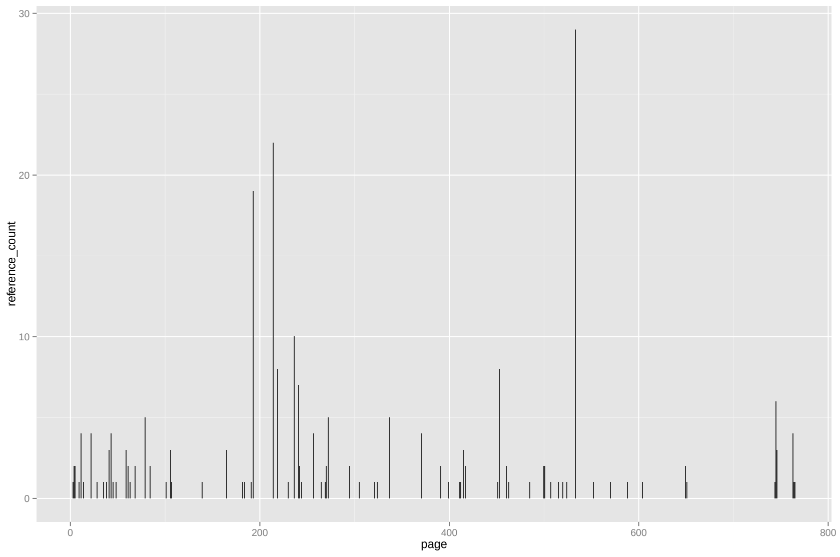


Figure 2.1: Pages cited from *Elements of Statistical Learning* by academic publications in all fields.

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 semiotic machinery of the book relies on vectors and orbital trajectories that converge from many different directions. 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 predictive models ranging across half a century. The reference list beginning at page 699 (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 documents in the humanities or social sciences, and, almost before anything else, prompts attention to the problem of reading

Barad, Karen!on experimental apparatus

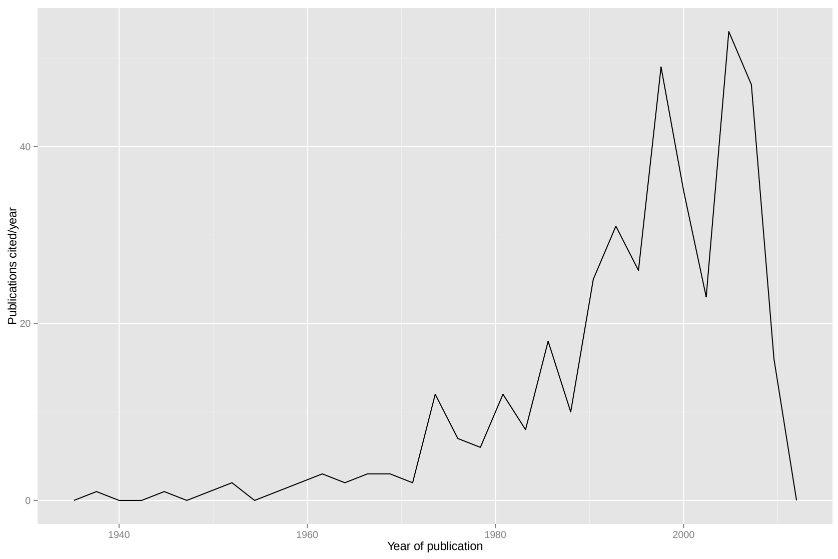


Figure 2.2: Publications cited by *Elements of Statistical Learning*. The references range over almost 80 years, with peaks in late 1970s, late 1980s, mid-1990s and mid-2000s. These peaks relate to different mixtures of cybernetics, statistics, computer science, medicine, biology and other fields running through machine learning. The smoothing of these peaks derives from use of local regression. Regression-related publications form the main body of citation.

parts and fragments, and relating to an object or perhaps an apparatus in the sense that Karen Barad ascribes to experimental setups (Barad 2007).

As Figure 2.2 indicates, 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. While many of these references either point to Hastie, Tibshirani or Friedman's own publications, or that of their statistical colleagues, the references show that these authors are also roving 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. Both the inward and outward movements of citation suggest that the book, like much in field of machine learning, has a wave function or matrix-like character that constantly superimposes and transposes elements across boundaries and barriers. The implication here is that machine learning as a field 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 we phronetically relate to it. . Hence, 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, biology is the topic of chapter **??** and optimization functions of chapter 3).

Kuhn, Thomas
phronesis
linear regression model|(

The obdurate mathematical glint of machine learning

While references from many different places go in and out of *Elements of Statistical Learning*, they are nearly all articulated in mathematical form. The mechanics of machine learning as a level of abstraction are mathematical, and these mechanics soon begin to operate in the vast diagram the book presents. Given that mathematics is itself diverse and multi-stranded, what kind of mathematics matters most here? The predictive models in *Elements of Statistical Learning* are dominated by a single prediction 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. (We have already seen something of this model in the Introduction, in the equations for linear regression **??**). The linear regression model pushes up some of the citational peaks in Figure 2.2. Even though it is an old technique dating back to Francis Galton in the linear regression model

1890s (see (Stigler 1986, chapter 8)), drawing sharp, straight lines through opaque clouds of data-matter is what much machine learning still seeks to do. It manoeuvres in complex ways in drawing these lines (as we will see), but lines cutting, cleaving, and shattering things are very much the central working abstraction of machine learning. Hastie, Tibshirani and Friedman acknowledge 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, linear regression is subjected to countless variations, iterations, expansions and modifications. Their own research spans several decades and a range of topics concerning linear regression models and their variations ('ridge regression'; 'least angle regression'; etc.). But this introduction of the 'mainstay of statistics,' the linear model, already introduces a harsh form – 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 2.1.

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j \tag{2.1}$$

In its relatively compressed typographic weave, expressions such as Equation 2.1 operationalize movements through data that call for some attention. These expressions, which are not comfortable reading by and large 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 other places. While their presence distinguishes machine learning from many much computer science where mathematical equations are less common, these equations also allow the book to collate and borrow from a panoply of scientific publications and datasets in fields of statistics, artificial intelligence and computer science. Along with the citations, the graphical plots, the algorithms (implemented in code described below), these equations are integral connective tissue in machine learning. Unless we come to grips with their diagram force, without succumbing to the obscuring dazzle of mathematical abstraction, the connectivity and mobility of these forms will be lost on us.

linear regression model|)
Peirce, C.S.
diagram|(
diagram!icon

I find it useful here to follow Charles Sanders Peirce account of mathematics in terms of diagrams. 'Mathematical reasoning,' he writes, 'is diagrammatic' (Peirce 1998, 206). That it, we should see mathematics, whether it takes an algebraic or geometrical form, whether it appears in symbols, letters, lines or curves as diagrams. Now 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 2.1? As Peirce says, 'many 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 will soon see, 2.1 could be expressed in statements in a programming language like `Python` or `R`, or in algorithmic pseudo-code, or perhaps most accessibly, as graphic figure (a line drawn through a cloud of points). In none of these associated diagrams can the relations between the parts be observed in the same way as they 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. The diagram!indexical sign

graphic form of the expression include the various classical Greek symbols such as $\sum$ or $\prod$, as well as the letters x, y, z and the indices (indexical signs) that appear in subscript or superscript, as well as the spatial arrangement of all these in lines and sometimes arrays. A variety of relations run between these different symbols and spatial arrangements. For instance, in all such expressions, the difference 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 2.1, 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 general 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 such diagrams suppress so many details, they allow one to focus on a more limited range of relations between parts. The manipulation of those relations generates new diagrams or patterns. This affordance of diagrammatic forms is extremely important in the intensification of machine learning. Importantly, diagrams can diagram other diagrams. Put differently, operations can be themselves the subject of operations. Or functions can themselves for functions of functions. This nesting and coiled aspect of the diagrams is highly generative since it allows what Peirce calls 'transformations' (212) or the construction of 'a new general predicate'(Peirce 1992, 303).[2] The intensive

2. Felix Guattari makes direct use of Peirce's account of diagrams as icons of relation

processing of data today via predictive models is largely channelled via such diagrams. These diagrams are not conspicuous in the infrastructures, and they are not directly seen by people or things they impinge upon even as they have their effect.

diagram|)

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 transparently obvious. Just as much as the analysis of a photograph, a literary work or an ethnographic observation, their semiotic movement 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 is well worth it if in consequence we can diagrammatically understand 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.

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 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.

Ng, Andrew

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 repay re-reading). The cost of the diagrammatic density of its 'elements' (equations, citations, tables, datasets, plots) is a certain refractory feeling of 'not quite understanding' for many readers. This feeling is inevitable because the book condenses finished work from several disciplines, and partly because it seeks to frame a great diversity of materials *statistically.* (The statistical aspects of machine learning are the main topic of chapter **??**). Professor 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).[3] Note that Ng is a computer scientist, not a statistician. 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

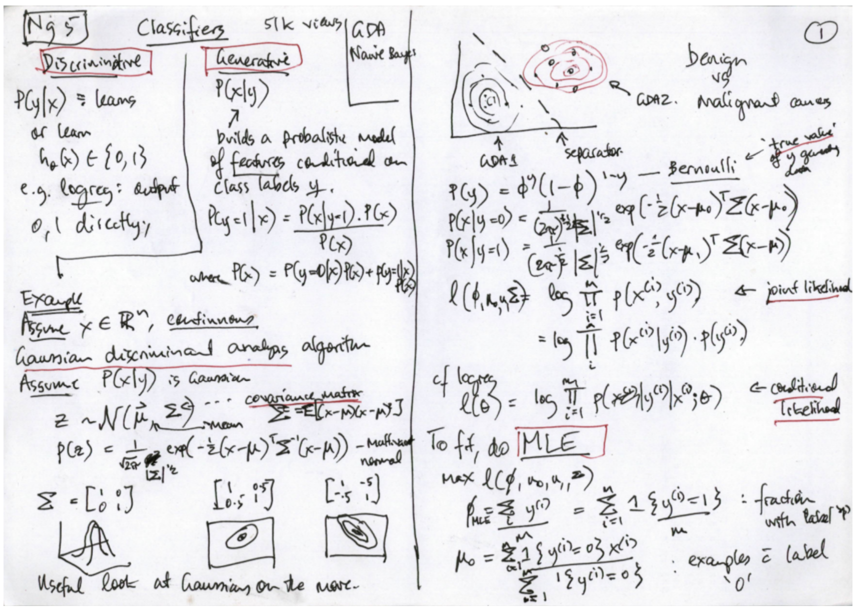
3. A heavily shortened version of this course has been delivered under the title 'Machine Learning' on Coursera.org, a MOOC (Massive Open Online Course) platform.

datasets and the simplest possible statistical models and machine learning algorithms (usually 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). Ng's CS229 lectures differ from almost all other expository materials in that we see someone writing and drawing 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.[4]

linear regression

Ng's Youtube anachronistic but popular lectures have a certain diagrammatic density that comes from the many hours of chalked writing he stages during the course. In a time when PowerPoint presentations or some other electronic textuality would very much have been the norm (2007), why is a Stanford computer science professor, teaching a fairly advanced postgraduate course, writing on a chalkboard by hand? (Figure 2.3 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 descriptive experiment, but more importantly an exercise 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 complied with Ng's injunction to 'copy it, write it

4. 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.



$\boxed{\text{Ng 5}}$ Classifiers 51k views

GDA

Naive Bayes

$\boxed{\text{Discriminative}}$

$P(y|x)$ ≑ learns

or learn

$h_\theta(x) \in \{0,1\}$

e.g. logreg: output

0, 1 directly;

$\boxed{\text{Generative}}$

$P(x|y)$

↗

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 $P(x) = P(y=0|x)P(x) + P(y=1|x)P(x)$

Example

Assume $x \in \mathbb{R}^n$, continuous

Gaussian discriminant analysis algorithm

Assume $P(x|y)$ is Gaussian

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

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

benign vs malignant cancers

GDA2.

GDA1

separator.

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

$\ell(\theta) = \log \prod_{i=1}^{m} p(~~x^{(i)}~~|y^{(i)}|x^{(i)};\theta)$ ← 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 2.3: Class notes lecture 5, Stanford CS229, 2007

out, cover it, and see if you can reproduce it' (**NgAndrew__2008**). While it occasions much writing and drawing, and many struggles to keep up with the diagrams that Ng narrates as he writes, it seems to me that this writing of equations, with all their substitutions and variations, alongside the graphic sketches of intuitions about the machine learning techniques, adds something of the *derivations* that is quite hard to finesse in *Elements of Statistical Learning.* There the diagrammatic weave between the expressions of linear algebra, calculus, statistics, and the algorithms is almost too tight to work with. In Ng's CS229 lectures, by contrast, the weaving, while still complex, is much more open. They lack the citational tapestry of (Hastie, Tibshirani, and Friedman 2009), 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 have a useful thawing effect. As we will see, only by tracking some of the movements of code that underpin machine learning do we identify levels of abstraction that we might want to think about.

diagram!hand-drawn

linear regression

Some broadly shared topic structures help in any navigation of these pedagogical 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 tradeoff).[5] 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. In this respect, Ng's step-by-step handcrafted derivations are too derivative to really track the trans-semiotic intensities of equations such as equation 2.1. How would one learn to read such diagrams less linearly (Ng's long columns of algebraic derivation), more diagonally and in better

5. 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.

alignment with the multi-faceted abstraction they often compact?

learning

Rosenblatt, Frank

machine
learner!perceptron|(

The learning of machine learning

For a book with 'learning' in its title, *Elements of Statistical Learning* has only a little to say explicitly 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 3, 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 deadend. There is no general, or comprehensive artificial intelligence in existence, even though many efforts continue to develop 'deep learners' (see for instance, Google Corporation's ongoing research into 'deep learning' of its own data; and chapter **??**). If a general intelligence did not appear, certainly a lot of learning was undertaken. The field of machine learning might be seen as one such offshoot since it paused the process of machines becoming intelligences at a developmental phase that entails much close supervision, monitoring and oversight.

The so-called 'learning problem' and the theory of learning machines developed largely 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, Rosenblatt implemented the perceptron, which today would be called a single-layer neural network (Hastie, Tibshirani, and Friedman 2009, 393) on a computer 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.[6] What has been *learned* in the intervening period?

If we compare the diagram of Rosenblatt's perceptron shown in Figure 2.4a to the typical contemporary diagram of a neural network shown in Figure 2.4b, 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 2.1. What has happened between the

6. Elizabeth Wilson's study, *Affect and Artificial Intelligence* (Wilson 2010), draws on a combination of psychoanalytic, psychological and archival materials discussing the work of key figures in the early history of artificial intelligence such as Alan Turing on intelligent machinery, Warren McCulloch and Walter Pitts on neural nets, and recent examples of affective computing and robots such as the MIT robot Kismet. Her framing of the psychic nexus with machines such as the perceptron is provocative:

> Sometimes machines are the very means by which we can stay alive psychically, and they can just as readily be a means for affective expansion and amplification as for affective attenuation. This is especially the case of computational machines (30).

Under what conditions do machines and for present purposes, computational machines, become 'the very means we can stay alive psychically'? Wilson addresses this question by positing 'some kind of intrinsic affinity, some kind of intuitive alliance between the machinic and the affective, between calculation and feeling' (31), and suggesting that the 'one of the most important challenges will be to operationalize affectivity in ways that facilitate pathways of introjection between humans and machines' (31). Introjection, the process of bringing the world within self is, according to psychoanalytic accounts of subjectivity, crucial to the formation of 'a stable subject position' (25). Wilson envisages introjection of machine processes as a good, not as a failure or attenuation of relation to the world. Note that her use of introjection differs strikingly from the sense of introjection invoked by Paolo Virno in his account of contemporary production (Virno 2004), but both Wilson and Virno retain a commitment to the primacy of psychic processes in the human-machine nexus. While I tend to go in the same direction as Wilson in relation to affective expansion, I don't tend to see that expansion as unfolding from introjection, but rather from an intensification of diagrammatic processes.

Vapnik, Vladimir

machine
learner!perceptron|)

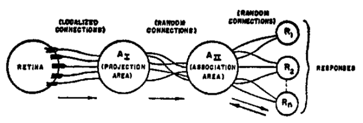
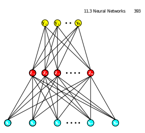


FIG. 1. Organization of a perceptron.

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



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

Figure 2.4: 1958 perceptron and 2001 neural net compared

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'[7].

7. In describing the entwined elements of machine learning techniques, and citing various machine learning textbooks, I'm not attempting to provide any detailed history of their development. My account 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

neural net
diagrammatization
diagram|(

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

What the perceptron latches onto

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 levels of abstraction. Following Peirce, we might begin to see machine learning as a diagram-

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.

matic 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 diagram|)

Pearson, Karl

artificial intelligence

as the same things appears in different places.

One step in this diagrammatic dance moves from what we might called symbolic logical diagrams to statistical algorithmic diagrams. While many of the machine learning techniques I discuss have quite long statistical lineages (runnning back to the 1900s in the case of Karl Pearson's development of the still-heavily used Principal Component Analysis (Pearson 1901)) , machine learning techniques also owe much to a certain dissatisfaction with the classical AI understanding of intelligence as manipulation of symbols. Symbolic intelligence, epitomised by deductive logic or predicate calculus, was very much at the centre of many AI projects during the 1950s and 1960s (Dreyfus 1972; Edwards 1996). In this line of diagrammatization, certain privileged symbolic-cognitive forms are subject to a kind of statistical transformation. Take for instance once of the most common operations of the Boolean logical calculus, the NOT-AND or NAND function shown in table 2.2.

Table 2.2: The Boolean function NOT-AND truth table

X1	X2	X3	Y
0	0	0	1
0	0	1	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

For present purposes, these kinds of logical and symbolic diagrams have triple relevance. First, the spatial arrangement of the table is fundamental artificial intelligence

to not only machine learning and most contemporary data practice. (Transformations in tables are the main topic of chapter **??**.) That spatial form is diagrammatic in various ways, but principally through the horizontal and vertical relationships it installs. Most datasets come as tables, or end up as tables at some point in their analysis. Second, the elements or cells of this table are numbers. These numbers, 1 and 0 are the binary digits as well as the 'truth' values 'True' and 'False' in classical logic. The truth table for NOT-AND is in some ways typical of data tables because it contain numbers. These numbers are readable as symbolic logical propositions. Hence this table acts as a hinge between numbers and symbolic thought or cognition. (In (Hastie, Tibshirani, and Friedman 2009) most tables contain data in the forms of numbers rather than truth values.) Third, the NAND table in particular has an obvious operational importance in digital logic, since digital circuits of all kinds – memory, processing, and communication – comprise such logical functions knitted together in the intricate fabrics of contemporary calculation.[8]

The obviousness of the logical operation shown in the table stands out. It is hardly surprising to us at all. This looks like the kind of mechanistic calculation that computers 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 such operations inscribed in transistor circuits? Machine learning of such truth tables is not a typical operation today, but does illustrate something of the diagrammatic transformations that classical AI has undergone . If we return to the perceptron, with its neural intuition, how could it learn this kind of logic? It knows nothing of the logical calculus, but it can be induced to take on

8. 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.

a logical shape by training it. For instance, a perceptron that 'learns' the binary logical operation NAND (Not-AND) is expressed in twenty lines of python code on the Wikipedia 'Perceptron' page (*Perceptron* 2013). The code is followed by the output – a series of numbers – produced when it runs.

```
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)
outfile = open('./perceptron_weights.txt', 'w+')

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

out = 'weight1, weight2, weight3, error_count, iteration'
print >>outfile, 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 >>outfile, 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


outfile.close()
```

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 2.3: Iterative change in weights as a perceptron learns the NAND function

What does this code show or say? First of all, we should note the relative conciseness of the code vignette. In citing this code, I'm not resorting to a technical publication or scientific literature as such, or even to a software library or package, just to that popular yet incredibly heavily used epistemic resource, the Wikipedia page, and the relatively generic and widely used programming language `python`.[9] Whatever the levels of abstraction associated with machine learning, it is hardly ever hermetically opaque.

9. There is much to discuss about programming and code in machine learning. I return to that below. The important point for present purposes is that this is pretty much vanilla standard stand alone `python` code. There are no esoteric libraries or dependencies here. As is often the case with machine learning, the abstract machines are quite simple, but their potential relationality is complex.

machine learner!perceptron|(

While perceptrons and neural networks are the topic of later chapters, the typical features of this code as the implementation of a machine learning algorithm are the presence of 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. Much of the code here is much more familiar, generic programming. It defines variables, sets up data structures (lists of numerical values), checks conditions, loops through statements or prints results. Executing this code (by copying and pasting it into a python terminal, for instance) produces several dozen lines of numbers that are initially different to each other, but that gradually converge on the same values (see printout). 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. It is a typical machine learning algorithm, almost always included in machine learning textbooks and usually taught in introductory machine learning classes. Perhaps more strikingly than its persistence as an algorithm over half a century, the implementation of the whole algorithm in twenty lines of code on a Wikipedia page suggests the mobility of these methods. What required the resources of a major university research engineering laboratory in the 1950s can be re-implemented by a cut and paste operation between Wikipedia pages and a terminal window on a laptop in 2014. This is now a familiar observation, and perhaps not very striking at all. Again, what runs across all of these observations are the numbers that the algorithm produces. The NAND truth table has been re-drawn as a list of tuples or sets (see line 4 of the code that defines the variable `training_set`). The perceptron has learned the truth table by being given 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 values of the truth table. The algorithm exits it main loops (`while True:`) when there are no errors. The effect here is recursive: a model or algorithm implemented in digital logic has learned a basic rule of digital logic at least approximately. This transformation in learning style is symptomatic of the broader transformations that machine learning latches onto. The learning done in machine learning has few cognitive or symbolic underpinnings. It differs from classical AI in that it takes existing symbolic and, increasingly, signifying processes (such as the cat faces that `kittydar` tries find), and latch onto them diagrammatically. The diagrammatic dance between different algebraic, geometrical, algorithmic and tabular forms of expression is where the learning occurs.

machine
learner!perceptron|)
learning

At the same time, the perceptron algorithm produces numbers - `0.79999`, `2.0` – as weights. 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 numbers – and they truly abound in machine learning – triggers many different diagrammatic movements in *Elements of Statistical Learning* and like-minded texts (and these different movements will be discussed in chapters **??** and **??**). These numbers engender much statistical ratiocination (see chapter **??**), but here we need only note the distance between symbolically organised diagrams like the NAND truth table of Table 2.2 and the diagrams of a machinic configuration printed as weights in table 2.3.

Domingos, Pedro

machine learners!programs

Mol, Anne-Marie

practice

Machine learning implemented as program

Pedro Domingos refers to change from building programs to growing knowledge (see the epigraph to this chapter). Does this contrast help make sense of the perceptron's operations or *Elements of Statistical Learning* more generally? The final part of my reading protocol for multi-faceted abstractions in machine learning entails, I am suggesting, tracking the transformations between table 2.2 and table 2.3 in and by program code. What would we learn by studying and implementing machine learners as programs rather than their history or the controversies associated with them? It is not the case that code offers privileged access to abstraction. But it is highly diagrammatic and in sometimes latent ways, affectively partipatory. That is, it expresses the multi-faceted abstraction of machine learning and allows many connections running across facets to be followed. Science studies scholars such as Anne-Marie Mol have urged the need to keep practice together with theories of what exists. Towards the beginning of *The Body Multiple: Ontology in Medical Practice* (Mol 2003), Mol writes:

> If it is not removed from the practices that sustain it, reality is multiple. This may be read as a description that beautifully fits the facts. But attending to the multiplicity of reality is also an act. It is something that may be done – or left undone (6)

Mol's work offers a cogent case for developing accounts of what is real steeped in the practices that make it real. While similar sounding affirmations of the underpinning role of practice can be found in many parts of social sciences and humanities (since who would not affirm the centrality of practice?), Mol's insistence on this nexus of practice or doing and the existence of things in their plurality offers another way forward in reading *Elements of Statistical Learning.* Tracking techniques – a more stabilised form of practice

– and the flow of their implementations is a way of keeping abstractions and calculations in their faceted multiplicity. The media theorist Ian Bogost speculates and practices coding as a way to make sense of the world: 'source code itself often offers inroads in alien phenomenology - particular when carpentered to reveal the internal experience of withdrawn units' (Bogost 2012, 105) . In relation to machine learning, reading and writing code alongside scientific papers, Youtube lectures, machine learning books and competitions is not only a form of observant participation, but directly forms part of the diagrammatic multiplicity. We can see this in a number of different ways.

Bogost, Ian
source code
programming language!R|(

First, although barely a single line of code appears in *Elements of Statistical Learning*, nearly all of the examples in *The Elements of Statistical Learning* are implemented in a single programming language, `R`. The authors say 'we used the R and S-PLUS programming languages in our courses' (Hastie, Tibshirani, and Friedman 2009, 9) but the diagrammatic movements of the book owe much more to `R` code.[10] The proliferation of programming

10. In a latter 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 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

programming languages

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 implementation of machine learning techniques could proceed. 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. They have 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.

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

Second, particular code elements such as `R` packages relate to many others, and these relations can be used to identify how different facets and levels of abstraction come together. For instance, the line of code shown above when executed opens another way of reading *Elements of Statistical Learning* and getting a feel for the dragnet of practice running through it. There is nothing too opaque, I would suggest, about the line of code itself, but there are some folds and code-specific convolutions associated with it that point to infrastructural undersides. Take the part of the line `dependencies = 'Suggests'`. When the line of code executes, this stipulation of `dependencies` leads to a quite wide-ranging installation event. If the installation works (and that assumes quite a lot of configuration and installation work has already taken place; for instance, installing a recent

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.

Marx, Karl

	How often suggested
testthat	220
knitr	150
MASS	67
testthat, knitr	46
RUnit	37
parallel	28
lattice	25
knitr, testthat	22
ggplot2	17
R.rsp	16
survival	16
mvtnorm	15
testthat, devtools	15
rgl	12
xtable	10
knitr, rmarkdown	9
testthat, roxygen2	9
akima	6
coda	6
scatterplot3d	6

Table 2.4: R packages suggested by the ElemStatLearn package

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 2.5 and **??**).[11] 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

11. 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 7356 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* (W. N. Venables and Brian D. 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.

	Package	How often depended on
methods	methods	73
stats	stats	28
MASS	MASS	26
survival	survival	26
mvtnorm	mvtnorm	21
Matrix	Matrix	13
ggplot2	ggplot2	9
grid	grid	8
igraph	igraph	8
lattice	lattice	8
rJava	rJava	8
rgl	rgl	7
tcltk	tcltk	7
XML	XML	7
utils	utils	6
nlme	nlme	5
ape	ape	4
coda	coda	4
graphics	graphics	4
gtools	gtools	4

Table 2.5: ‘R‘ packages depended on by the ‘ElemStatLearn‘ package

packages in online repositories suggest that a highly specialised division of labour and possibly refined co-operative labour processes operate around data.

Third, `R` is an increasing well-known and widely used statistical programming language and environment (R Development Core Team 2010). 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 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 2010). 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.

Chambers, John

Fourth, less pragmatically, but symptomatically, R has also attracted mainstream media attention. An article in *The New York Times* in 2009 highlighted its practical importance in data analysis (Vance 2009), at the time *The New York Times* was heavily promoting its idea of data journalism. This is somewhat unusual, as programming languages are not normally the topic of mainstream media interest. It is easy today to find an employment-centric view of data practice. [12] `R` is an evocative object, to use the psychoanalyst

12. The 'chief economist' at Google, Hal Varian in 2009 made a widely quoted prediction about working with data: 'The ability to take data — to be able to understand it, to process it, to extract value from it, to visualize it, to communicate it — that's going to be a hugely important skill in the next decades, not only at the professional level but even at the educational level for elementary school kids, for high school kids, for college kids' (McKinsey 2009).

While cited here in a report prepared by the global business consultancy, McKinsey & Co, this quote, or selections from it, can be seen in many different contexts, ranging from government reports on higher education to student noticeboards in university statistics departments. What Varian, with all the allure of Google as a potential employer, presents as an 'important skill' also has a 'scarce factor': the 'ability to understand that data.' While Varian presents understanding data as the prelude to 'extract[ing] value from it,' understanding might take different forms, depending on the kind of encounter or engagement that precedes it.

A somewhat broader framing of the value of `R` can be found voiced as a form of democratised knowing or engagement with data. Norman Nie, the original developer of the widely-used SPSS social science statistics software (see (Uprichard, Burrows, and Byrne 2008)), is a proponent of `R`. Norman Nie interviewed in *Forbes*, a leading business news magazine, evangelises for `R`: 'Everyone can, with open-source `R`, afford to know exactly the value of their house, their automobile, their spouse and their children–negative and positive … It's a great power equalizer, a Magna Carta for the devolution of analytic rights' (Hardy 2010).

Nie, the founder of Revolution Analytics, a company that provides software and support services to `R` users, promotes `R` as a way of protect individual liberties to know what they own. In the context of the global financial crisis of 2007, it may be that the value of houses became much more problematic, and understandably, harder to know. And the reference to 'spouse and their children' must be flippant. Yet the 'devolution of analytic rights' that Nie speaks of here, even as he frames it in terms of entrepreneurial individualism, is important. While Nie's company Revolution Analytics promotes the incorporation of `R` into powerful and pervasive business and government prediction systems (such as SAS and IBM's Pure Data (IBM 2013)), Nie here points to the problem of 'manipulation and control' of customers associated with just that incorporation. The 'analytic rights' he advocates in the form of `R` are meant to help individuals counterbalance the power of large scale data analysis conducted by corporations and governments. More than any particular or specific use, Nie's advocacy of `R` taps into a wider sense of rich possibility associated with `R`. It would be possible to cite many other instances of this belief and desire in the potential of `R`. A good indication of this overflow is the aggregate blog, R-bloggers. `R`-bloggers brings together several hundred `R`-related blogs in one place. The range of topics discussed there on any one day – Merrill-Lynch Bond Return indices, how to run `R` on an iPhone, using US Department of Agriculture commodity prices, analysing human genetic variation using PLINK/SEQ via `R`, using the 'Rhipe' (sic!) package to program big data applications on Map-Reduce cloud architectures, doing meta-analysis of phylogenetic trees, etc., (R-bloggera 2011) – euggest how widely desires/beliefs in `R` range. To take just one fairly recent example, '`R` Analysis Shows How UK Health System could save £200 million' claims a recent post on the site (D. Smith 2012). While many of the applications, experiments, or techniques reported there can be reduced to employment or to individual uses of `R`, they also, as we will see, suggest the potential for encounters and engagements. Something similar, and perhaps less bound to commercial data analytics can be found in Rachel Schutt and Cathy O'Neil's much more lively *Doing Data Science* (Schutt and O'Neil 2013) as it discusses the ethics and potentials of working with machine learning models and programming languages such as `R` and Python.

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 intersect in `R`. Bollas suggests that 'our encounter, engagement with, and sometimes our employment of, actual things is a *way* of thinking' (92). `R` embodies something of plurality of practices of working with measurements, numbers, text, images, models and equations, with techniques for sampling and sorting, with probability distributions and random numbers. 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.

Bollas, Christopher
thinking
Ripley, Brian
Venables, Bill

Writing code has always been central in machine learning where algorithms and machines are the primary expressive forms that ideas take as they become diagrams. Two of the main 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. (W. Venables and B. D. Ripley 2000, 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 hyperobject!machine learning as

'software' here not in the sense of a product, but in the sense that today would more likely be called 'code.' This sense of coding and programming as clarifying and concretising ideas with precision has thoroughly taken hold in contemporary data analysis. But not that it lies somewhat at odds with Pedro Domingos characterisation of machine learning as a shift away from building to growing. Many practices, many ways of doing things, describe themselves as part of their doing. Perhaps every practice contains elements of its own description. The contrast between `R` as a language originating in statistics and subsequently percolating through various scientific fields and `Python` or `Java`, which are much more general purpose programming languages widely used for software development in many settings, is instructive. Implementations of machine learning algorithms in `Python` can more or less be found in a few software libraries such as `SciPy` and `ScikitLearn` (Pedregosa et al. 2011), and much of the instructional materials on how to use machine learning in practice rely on `Python`. But these implementations are somewhat less diagrammatically rich than those associated with `R`. They tends to be operational rather epistemic.

The diagram of a hyperobject

I have been suggesting that we can get a sense of the hyperobject-like character of machine learning by treating *Elements of Statistical Learning* as a diagram of operations or more simply, and as an index of the operations of machine learners in publication, in computation, in code. Whether mapped through research publications, pedagogical materials or code libraries in `R`, these operations move across and connect facets of abstraction that include, but are not limited to, mathematical forms (especially statistical ones, but also algebra and calculus), problems from multiple scientific disciplines (especially computer science, but also biology, medicine and others), devices such as computing platforms, data formats and algorithmic operations and pieces and parts of code. Following Peirce, we can treat all of these elements as diagrammatic components. Diachronically, or in time, machine learners have patched these elements together in ways that have begun to substantially affect what counts as knowledge, action, or decision. The operational power of machine learning does not stem from a single layer of abstraction. Hence any engagement with machine learning and similar operations would be very limited if it confined its range of movement to a single level such as code, or algorithm or mathematical function or platform. 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.

table

machine learners!variety of|(

Domingos, Pedro

Chapter 3

Machines finding functions

> Because of a gradient that no doubt characterizes our cultures, discursive formations are constantly becoming epistemologized (Foucault 1972, 195)

The opening pages of machine learning textbooks often warn or enthuse about the profusion of techniques, algorithms, tools and machines. 'The first problem facing you', writes Pedro Domingos, 'is the bewildering variety of learning algorithms available. Which one to use? There are literally thousands available, and hundreds more are published each year (Domingos 2012, 1). 'The literature on machine learning is vast, as is the overlap with the relevant areas of statistics and engineering' writes David Barber in *Bayesian Reasoning and Machine Learning*(Barber 2011, 4); 'statistical learning refers to a vast set of tools for understanding data' writes James and co-authors in an *Introduction to Statistical Learning with R* (James et al. 2013, 1); or writing in in *Statistical Learning for Biomedical Data* the biostatisticians James Malley, Karen Malley and Sinisa Pajevic 'freely admit that many machines studied in this text are somewhat mysterious, though powerful engines' (Malley, Malley, and Pajevic 2011, 257). In *Thoughtful*

Kirk, Matthew

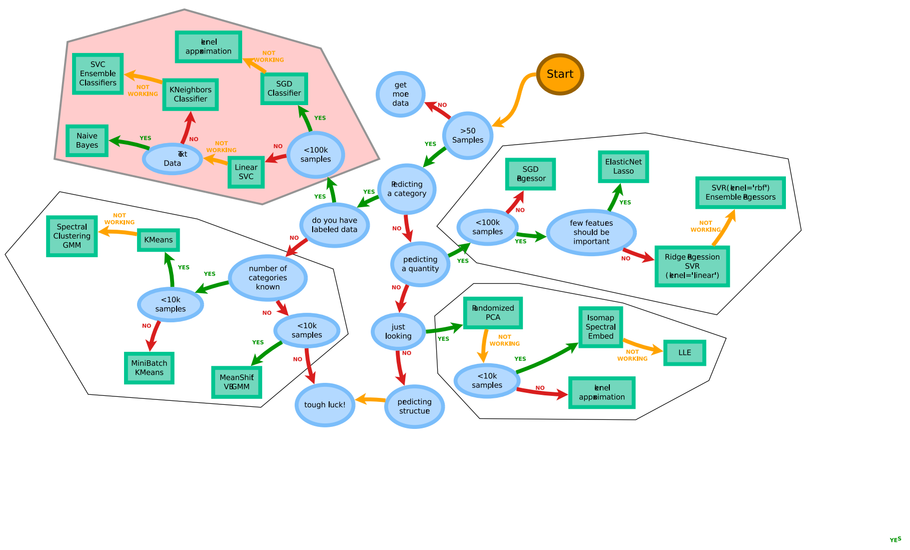


Figure 3.1: ‘scikit-learn‘ map of machine learning techniques [TBA: ref to diagram]

Machine Learning Matthew Kirk exacerbates the situation: ‘flexibility is also what makes machine learning daunting. It can solve many problems, but how do we know whether we’re solving the right problem, or actually solving it in the first place?’ (Kirk 2014, ix). The prefatory comments from Domingos, Barber, James, Malley and Kirk suggest a rampant even weedlike abundance of machine learners, as does the 700 or so pages of *Elements of Statistical Learning*. Much learning of machine learning work, at least for many practitioners, concerns not so much implementation of particular techniques (neural network, decision tree, support vector machine, logistic regression, etc.), but rather navigating the maze of methods and variations that might be relevant to a particular situation.

How does this effect of accumulation and profusion arise? And what do machine learners do about it? The publications I have just been referring to present that profusion as a problem of the piling up of scientific publications. As the authors of textbooks and how-to-manuals, they attempt to manage it by providing, indexes, maps and guides to the bewildering variety of machine learners. *Elements of Statistical Learning* deploys tables, overviews, theories of statistical modelling, model assessment and comparison techniques to aid in navigating them. Parallel and complementary mappings accompany software libraries. The visual map of machine learning techniques shown in Figure 3.1 comes from a particular software library written in `Python`, `scikit-learn` (Pedregosa et al. 2011). This software library is widely used in industry, research and commerce. In contrast to the pedagogical expositions, theoretical accounts or guides to reference implementation, code libraries such as `scikit-learn` tend to order the range of techniques by offering recipes and maps for the use of the *functions* the libraries supply. The branches in the figure lay down paths through the profusion of techniques as a decision tree.[1] Similarly, for `R` code, the *Comprehensive R Archive Network* tabulates key libraries of `R` code in a machine learning 'task view' (Hothorn 2014).

Python!scikit-learn library

function

machine learners!variety of|)

programming language!R!Comprehensive R Archive Network

```python
import sklearn
from sklearn import *
modules = dir(sklearn)
modules_clean = [m for m in modules if not m.startswith('_')]
print('\n'.join(modules_clean))
```

/usr/local/lib/python2.7/dist-packages/sklearn/pls.py:7: DeprecationWarning: This module has been moved to cross_decomposition and will be removed in 0.16 "removed in 0.16", DeprecationWarning) base clone cluster covariance cross_decomposition cross_validation datasets decomposition dummy ensemble externals feature_extraction feature_selection gaussian_process grid_search hmm isotonic kernel_approximation lda learning_curve linear_model manifold metrics mixture multiclass naive_bayes neighbors neural_network pipeline pls preprocessing qda random_projection re semi_supervised setup_module svm sys test tree utils warnings

1. See Chapter ?? for discussion of decision trees in machine learning.

abstraction!levels of

function!different senses of!mathematical, algorithmic, operational

The architecture of these software libraries itself classifies and orders machine learners. `Scikit-learn` for instance comprises a number of sub-packages: Modules such as `lda` (linear discriminant analysis), `svm` (support vector machine) or `neighbors` (k nearest neighbours) point to well-known machine learners, whilst `cross-validation` or `feature_selection` refer to ways of testing models or transforming data respectively. Again, machine learning patches together a variety of abstractive practices. These divisions, maps and classifications help order the techniques, but they obscure the problematic process that first generated a competing profusion of machine learners. That profusion comes from a slippage between three main senses of the function: function as mathematical relation or operator, function as concrete machinic operation and function in the sense of what something does.[2]

All three senses of function constantly coalesce and hybridize in machine learning research. Table 3.1 shows the titles and author-supplied keywords of a sample of well-cited machine learning publications. In these randomly chosen publications, mathematical functions – 'kernel function,' 'discriminant function,' 'radial basis function' – mingle with biological and engineering functions – 'protein-binding function,' 'intestinal motor function', or 'rules to control locomotion.' Mathematical functions, however, dominate here. Machine learners 'find', 'estimate,' 'approximate,' 'analyse' and sometimes 'decompose' mathematical functions. The primary mathematical sense of a function refers to a relation between sets of values or variables. (A variable is a symbol that can stand for a set of numbers or other values.) A function is one-to-one relation between two sets of values. It maps a set of arguments (inputs) to a set of values (outputs, or to use slightly more technical language, it maps between a *domain* and a *co-domain*.) As we

2. I discuss the sense of function as operation or process in chapter ??. There I suggest that this important sense of function as operation or process, a sense that has underpinned transformations in life, social and clinical sciences may be shifting towards a different ordering.

have already seen, mathematical functions are often written in formulae of varying degrees of complexity. They are of various genres, provenances, textures and shapes: polynomial functions, trigonometric functions, exponential functions, differential equations, series functions, algebraic or topological functions, etc. Various fields of mathematics have pursued the invention of functions. In machine learning and information retrieval, important functions would include the logistic function (discussed below), probability density functions (PDF) for different probability distributions (Gaussian, Bernoulli, Binomial, Beta, Gamma, etc. See chapter **??**), and error, cost, loss or objective functions (these four are almost synonymous). cost function!see{function!cost} The latter group I discuss below because they underpin many claims that machine learners learn.

probability!distribution
function!mathematical
\
diagrammatic!diagonal

From an *almost* purely mathematical standpoint, machine learning can be understood as function finding operations. Implicitly or explicitly, machine learners find a mathematical expression – a function – approximating an outcome of the social, technical, financial, transactional, biological, brain, heart or group process that generated the data in question. Regardless of the application, no single mathematical function perfectly or uniques expresses data. Many if not infinite functions can approximate any given data. This abundance of functions in some ways makes learning harder. The mathematical function and the software function are in diagrammatic relation, but the *diagonals* that run between them are not always in a one-to-one mapping.

Supervised or unsupervised, who learns what?

The techniques, models, forms of abstraction, data formats, and performance properties of machine learners have to be learned by people (reading books, common vector space

machine learning!learning|(

attending classes, watching demonstrations, trying out software and code, etc.; see Chapter ??). The `scikit-learn` map of techniques shown in Figure 3.1 addresses the problem of choosing a machine learner. This map is only a starting point. No matter how powerful machine learners become, they do not operate autonomously. Once learned, maps such as the one shown in Figure 3.1 may become redundant. But other forms of close attention, monitoring, and observation remain crucial whenever machine learners encounter data or wherever they enter the common vector space.

A need to observe the machine organises the field of machine learning. The optics of this observation of machine learners traversing common vector space vary, but they are always partial or incomplete. Machine learning textbooks and courses usually distinguish 'supervised', 'unsupervised' and sometimes 'semi-supervised' learning. While the field is almost despotically pragmatic in its commitment to optimisation of classification and prediction (although in certain ways, curiously idealistic too in its constant reuse of well-worked datasets such as `iris` or `South African heart disease`), it maintains the difference between two broadly different kinds of *learning*. Writing around 2000, Hastie et. al. state:

> With supervised learning there is a clear measure of success or lack thereof, that can be used to judge adequacy in particular situations and to compare the effectiveness of different methods over various situations. Lack of success is directly measured by expected loss over the joint distribution $Pr(X, Y)$. This can be estimated in a variety of ways including cross-validation. In the context of unsupervised learning, there is no such direct measure of success. ... This uncomfortable situation has led to heavy proliferation of proposed methods, since effectiveness is a matter of opinion and cannot be verified directly. (Hastie,

Tibshirani, and Friedman 2009, 486-7)

data!training

data!test

partial observer|seefunction!as partial observer

Supervised learning in general terms constructs a model by training it on some sample data (the training data), and then evaluating its effectiveness in classifying or predicting test data whose actual values are already known. The 'clear measure of success' they refer to in relation to in so-called 'supervised learning' is of relatively recent date.[3] Unsupervised machine learning techniques generally look for patterns in the data without any training or testing phases (for instance, k-means or principal component analysis do this, and both techniques have been heavily used for more than fifty years). In both supervised and unsupervised learning people look at the models to find out how the models traverse, fit, partition or map the data. At a general level, machine learning is a system of making statements and rendering relations visible through supervision. As I will suggest below, partial observers – a term drawn from the work of Félix Guattari and Gilles Deleuze – supervise the diagrammatic functioning of machine learning. At the same time, opacity – 'no direct measure of success' – is generative in machine learning. Amidst the optically dense pages of mathematical functions, plots of datasets and listing of algorithms, *Elements of Statistical Learning*'s frank admission that something cannot be measured and that this difficulty has led to proliferating methods seems to me quite promising ground to explore for possible transformations and changes. But this discomfort about unsupervised learning does seem to discourage its use. If, as the first part of the quoted text puts it, supervised learning has a clear 'measure of success,' that success only seems to encourage further variations and comparisons that end up proliferating machine learners, their publications and their software implementations. Levels of predictive success may vary widely with different techniques and different situations, but an

3. Only in the mid-1980s were the first theories of algorithmic learning formalised (Valiant 1984).

machine learning!learning|)
function!mathematical|(
Breiman, Leo

almost unbounded optimism associated with machine learning (for instance as more or less unspoken foundation of any analysis of 'big data') runs pell-mell across all of them.

Which function operates?

Formally, the differences between machine learners appear as mathematical functions. The mathematical sense of function is writ large in nearly all machine learning literature since functions diagram the relations on which devices and machines operate. Indeed, we might say that machine learning is nothing other than a machinic version of the functions that have long interested and occupied mathematicians. Importantly, functions support both the operations and the ordering of those operations. Classifiers, or machine learners that allocate case to categories, are often identified directly with functions:

> A classifier or classification rule is a function $d(\mathbf{x})$ defined on $\mathscr{X}$ so that for every $\mathbf{x}$, $d(\mathbf{x})$ is equal to one of the numbers $1, 2, ..., J$ (Breiman et al. 1984, 4)

Writing in the 1980s, the statistician Leo Breiman describes classifiers – perhaps the key technical achievement of machine learning and certainly the catalyst of many applications of machine learning – in terms of functions. A classifier *is* a function $d(\mathbf{x}$ where is $\mathbf{x}$ is the data and d ranges over numbers that map onto categories, rankings or or other forms of order and belonging. The equation of machine learners to functions is quite pervasive. Learning, predictions, and the classifications produced by machine learning derive from functions. The identification of machine learning with functions appears in the first pages of most machine learning textbooks. Learning in machine learning means finding a function that can identify or predict patterns in the data. As *Elements of Statistical Learning* puts it,

Vapnik, Vladimir

> our goal is to find a useful approximation $\hat{f}(x)$ to the function $f(x)$ that underlies the predictive relationship between input and output (Hastie, Tibshirani, and Friedman 2009, 28).

In a highly compressed form, this statement of goals doubles the function. It contains *the* function that generated the data as a foundation. This function figures as a ground truth almost physically or existentially imputed to the world. It also refers to 'finding ... $\hat{f}(x)$', where the '$\hat{}$' indicates an approximation produced by an algorithmic implementations, and it avers to 'use'. Similar statements pile up in the literature. Perhaps more importantly, *learning* here is understood as function finding. A leading theorist of learning theory Vladimir Vapnik again uses the language of approximation: 'learning is a problem of *function estimation* on the basis of empirical data' (Vapnik 1999, 291).[4] The use of the term 'learning' in machine learning displays affiliations to the field of artificial intelligence, but the attempt to find a 'useful approximation' – the 'function-fitting paradigm' as (Hastie, Tibshirani, and Friedman 2009, 29) terms it – stems mainly from statistics. Not all accounts of machine learning emphasise learning as function fitting. Some retain the language of intelligent machines (see for example, (Alpaydin 2010, xxxvi) who writes: 'we do not need to come up with new algorithms if machines can learn themselves'). Despite any differences in the framing of the techniques, all accounts of machine learning, even those such as *Machine Learning for Hackers* (Conway and White 2012) that eschew any explicit recourse to mathematical formula, rely on the formalism and modes of thought associated with mathematical functions.

4. Vapnik is said to have invented the support vector machine, one of the most heavily used machine learning technique of recent years on the basis of his theory of computational learning. Chapter **??** discusses the support vector machine.

function!mathematical|) Whether they are seen as forms of artificial intelligence or statistical models, the formalisms are directed to build 'a good and useful approximation to the desired output' (Alpaydin 2010, 41), or, put more statistically, 'to use the sample to find the function from the set of admissable functions that minimizes the probability of error' (Vapnik 1999, 31). Note the unobtrusive but crucial caveats in this formulation: functions must be found from a set of 'admissable functions.' Functions anchor machine learning so that we don't have to come up with algorithms, yet functions themselves have to be found under certain constraints.

Which functions does machine learning admit? As is often the case in working with a massive technical literature, the first problem in making sense of what is happening with function in machine learning concerns sheer abundance. The pages of (Hastie, Tibshirani, and Friedman 2009) are marked with score of references to 'functions': quadratic function, likelihood function, sigmoid function, loss function, regression function, basis function, activation function, penalty functions, additive functions, kernel functions,step function, error function, constraint function, discriminant function, probability density function, weight function, coordinate function, neighborhood function, and the list goes on. This list stands against a background of several hundred mathematical functions commonly used in science and engineering.[5] Clearly we cannot expect to understand the functioning of all these functions in any great detail. However, even a glance through this prickly list of terms begins to confirm the heavy reliance on functions in this field (as perhaps in many other science and engineering disciplines), but that the association between functions might be a way to

5. The U.S. National Institute of Standards published *The Handbook of Mathematical Functions* in 1965 (Abramowitz 1965). This heavily cited volume, now also versioned online lists hundreds of functions organised in various categories ranging from algebra to zeta functions. While a number of the functions and operations catalogued there surface in machine learning, machine learners implement, as we will see, quite a narrow range of functions.

map some important operations occurring in and around machine learning. We can also already see in this list that the qualifiers of the term function are diverse. Sometimes, the qualifier refers to a mathematical form – 'quadratic,' 'coordinate', 'basis' or 'kernel'; sometimes it refers to statistical considerations – 'likelihood', 'regression', 'error,' or 'probability density'; and sometimes it refers to some other concern that might relate to a particular modelling device or diagram – 'activation,' 'weight', 'loss,' 'constraint,' or 'discriminant.'

Stengers, Isabelle

function!learning|(

What does a function learn?

Functions configure the diagrammatic functioning of machine learning. In what sense does a function learn? The philosopher of science Isabelle Stengers seems to view functions pessimistically:

> No function can deal with learning, producing, or empowering new habits, as all require and achieve the production of different worlds, non-consensual worlds, actively diverging worlds (Stengers 2005, 162)

In some ways, Stengers would, on this reading, be taking a fairly conventional position on mathematical functions. They cannot learn or produce anything, only reproduce patterns that we already recognise. Similar statements might be found in many philosophical writings on science and on mathematics in particular.[6] But elsewhere in her writing Stengers explicitly

6. A major reference here would be Ernst Cassirer (Cassirer 1923) who posited a philosophical-historical shift from ontologies of substance reaching back to Aristotle's categories (Aristotle 1975) to a functional ontology emerging in 19th century as the notion of function was generalized across many mathematical and scientific fields. (See (Heis 2014) for a recent account of the *FunktionBegriff* in Cassirer's philosophy/) In a recent article, Paolo Totaro and Domenico Ninno suggest that the transition from substance to function occurs practically in the form of the algorithm (Totaro and Ninno 2014).

science!experiment

affirms *experimental practice*, much of which depends on functions and their operations (Stengers 2008). It might be better to say that she limits the claims made about the functions in order to highlight the specific power of science: 'celebrating the exceptional character of the experimental achievement very effectively limits the claims made in the name of science' (Stengers 2011, 376). (Limiting claims made for science might save it from being totally re-purposed as a techno-economic innovation system.)

The connection between a given function and a given concrete experimental situation is highly contingent or indeed singular. Stengers argues that mathematical functions impinge on matters of fact via a reference between a function and an experimentally constructed matter of fact:

> The reference of a mathematical function to an experimental matter of fact is neither some kind of right belonging to scientific reason nor is it an enigma, but actually the very meaning of an experimental achievement (Stengers 2005, 157).

The generic term 'reference' here harbours a multitude of relations. The experimental achievement, the distinctive power of science, works through a tissue of relations that connect people, things, facts and mathematical functions in a highly heterogeneous weave.[7] When a biomedical experiment uses *logistic regression* to 'estimate the probability that a critically-ill lupus patient will not survive the first 72 hours of an initial emergency hospital visit' (Malley, Malley, and Pajevic 2011, 5), they are doing machine

The idea of computable functions lies at the base of theoretical computer science and has been a topic of interest in some social and cultural theory (e.g. (Parisi 2013); see also my (Mackenzie 1997)), but Totaro and Ninno's suggest that algorithmic processes, as the social practice of the function, form contradictory hybrids with remnants of substance, in particular, categories and classification. . They see bureaucratic logic, for instance, as hopelessly vitiated by a contradiction between classification and function. Machine learning, I'd suggest, is an important counter-example. It hybridises function and classification without any obvious contradiction.

7. This point has often been made in the social studies of science; see (Latour 1993) for a very high-level account.

learning, and the value of their predictions is not captured by classical statistical approaches (analysis of variance, correlations, regression analysis, etc.). As machine learning techniques and the underpinning mathematics of probabilistic learning theory circulate more widely across different scientific disciplines (geography, ecology, astronomy, epidemiology, genomics, chemistry, communication engineering), in each setting the experimental achievement consists in constructing references between the mathematical functions and the matters of fact generated by instruments, observations and measurements and cantilevered by previous experiments. The question from Stengers' standpoint is this: what happens to the structure of referrals through experiments and the accumulated knowledge when functions are said to learn? In order to address this question, we need to delineate how functions function in machine learning. That could be in several different ways: as statements or utterances, as formula-diagrams, as graphic forms and in operational implementations as code. Any account of machine learning as function finding needs to map the concatenation of these different elements, none of which alone can anchor the 'learning' that goes on in machine learning.

function!learning
function!learning|)
diagrammatic!experiment

At first glance, machine learning as a field is not very experimental (even it radically influences the conduct of experiments in many scientific fields; see chapter **??**). It lacks the apparatus, the instruments, the laboratories, field sites or clinics of experimental practice. Experimentation, if there is any, takes place principally in the form of rendering diagrammatically visible the relays or referrals between different functions as they traverse data. While functions are often written in formulae, they are also diagrammed in graphic forms as plots and in operational forms as code or software. Many of the characteristic functions listed above appear in the graphic form of lines and curves. This diagrammatic entanglement of machine learners in function!mathematical!invention of

figure/hastie_2009_page_13.pdf

(a) Linear regression classifiered diagrammed through simulated data (Hastie, 2009, 13)

figure/hastie_2009_page_15.pdf

(b) A k-nearest neighbours classifier diagrammed through simulated data (Hastie, 2009, 15)

curves is not surprising. The historical invention of the term 'function' by the philosopher G.W. Leibniz in the 17th century relates to curves and their description. Functions for Leibniz describe variations in curves such as their slope. Identifying and locating these *singularities* in curves still preoccupies the function-finding done in machine learning. In contrast to the table, or the generalized common vector space that expands to accommodate all differences, the curve and the many graphic objects that seek to show curves in different ways, display continuous variation and singular points.

The 15-nearest neighbour classifier shown in Figure 3.2b layers simulated data (2 dimensions and an output variable with the values `BLUE` or `ORANGE` (Hastie, Tibshirani, and Friedman 2009, 12)) and a meandering line that classifies the simulated observations in terms of their class membership. Viewed in terms of their visual composition, many machine learning diagrams are organised around such lines that contour, divide, bound or surround the data captured from or emitted by the world. Curving lines outnumber all other graphic forms, whether the curves are the 'decision boundaries,' the plots of 'training errors,' the 'contours of constant density' (109), or the 'regularization path' for the South African heart data (126). There is a graphic!line and curve

function!diagram

constant tension in these graphic forms between line and curve. Consonant with the vectoral ideal of a straight line that either connects or cuts the data (as shown in Figure 3.2a), many of the graphics in the book (and others like it, especially the books written by computer scientists) show strong preferences for straight lines. But variations and uncertainties of various kinds detour the pursuit of the straight lines. Curves proliferate as machine learners try to the soften the rigidity of the lines or find regular paths for lines through irregular terrain. Viewed very naively, the contrast between lines and curves in the two figures above, a contrast replayed many times in *Elements of Statistical Learning*, suggests that different diagrammatic operations are at work around the data, sometimes aligning and straightening relations (for instance, the many linear models) and sometimes tracing much more non-linear paths through the data (for instance, as in k nearest neighbours or much more indirectly convolutional neural networks). In the several hundred colour graphic plots in (Hastie, Tibshirani, and Friedman 2009), a striking mixture of network diagrams, scatterplots, barcharts, histograms, heatmaps, boxplots, maps, contour plots, dendrograms and 3D plots respond to this tension between linearity and curvature.[8] Many of these graphic forms are common in statistics (histograms and boxplots), but some relate specifically to data mining and statistical learning (for instance, ROC – Receiver Operating Curve – or regularization path plots). A significant proportion of these graphics show no data from experiments or measurements, but nearly all of them either show the results of attempts to diagram a line or find a function (the two are almost synonymous) that refers the different data elements in the common vector space to each other.

8. A montage of all the graphics in *Elements of Statistical Learning* can be found at [TBC]

common vector space

Diagramming with curves: the logistic function

function!sigmoid|seealsofunction!logistic

function!logistic

Curves, in their spectrum of curvatures ranging from flat to intricately convoluted, refer functions to concrete situations. As we have already seen, machine learners often refer to 'fitting', as well as 'over-fitting' and 'under-fitting.' *Fitting* is a way of bringing functions into the data. As we saw in the previous chapter, the common vector space cannot be fully seen. Graphic plots and statistical summaries offer perspectival views on it, but machine learners traverse it by finding a mapping between the dimensions of the vector space. They do that by smoothing and aligning data. Take the example of *sigmoid* functions. These quite simple functions underpin many classifiers and animate many of the operations of neural network, including their recent re-incarnations in 'deep learning' (Hinton and Salakhutdinov 2006; Mohamed et al. 2011). A well-known example of a sigmoid function, the logistic function, can be written as:

$$f(x) = 1/(1+e^{-kx}) \tag{3.1}$$

The logistic function (shown as Equation 3.1 and as two curves in Figure 3.3), as we will see, is very important in many classification and decision settings precisely because of its *non_linear* shape and its constrained movement within a limited range of values (0 to 1). How does a function such as the sigmoid function 'fit' anything? Here the curve itself and even the name 'sigmoid' is the best guide. The S-shape of the sigmoid curve is a good guide to operations associated with curves. The logistic function has quite a long history in statistics since that curve diagrams growth and change in various ways. (As the historian of statistics J.S. Cramer writes: 'The logistic function was invented in the nineteenth century for the description of the growth of organisms and populations and for the

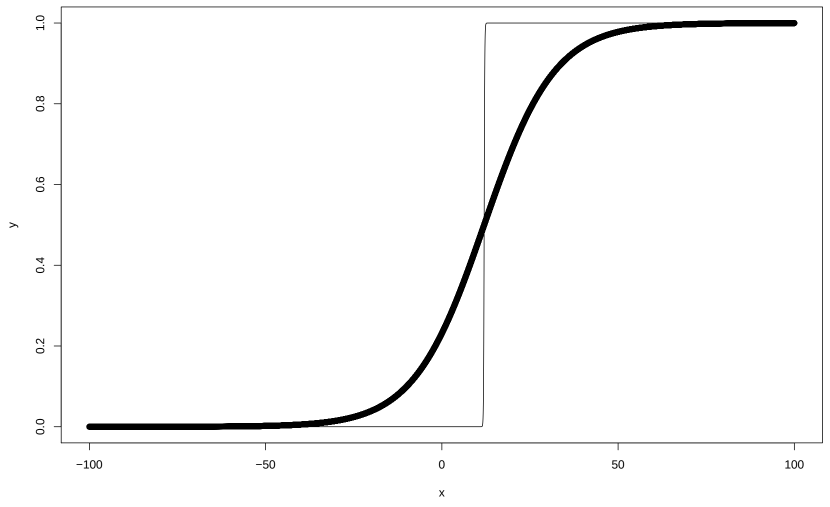


Figure 3.3: Logistic or sigmoid function

course of autocatalytic chemical reactions' (Cramer 2004, 614).[9] In nearly all of these cases, the function was used to fit a curve to data on the growth of something: populations, reactions, tumours, tadpoles tails, oats and embryos. The reference of the curve to growth comes from its changing slope. Growth starts slowly, increases rapidly and then slows down again as it reaches a limit. In the second half of the twentieth century, it was widely used in economics. In all these settings and usages, the curve was a way of summarising and predicting growth. Census data, clinical or laboratory measurements supplied the actual values of $f(x)$ at particular times, the x values. The task of the demographer, physiologist or economist was to work out the values of parameters such as k that controlled the shape of the curve. The logistic function had a well-established biopolitical resonance.

Note that the curves showing in Figure **??** plot the same data ($\mathbf{X}$ and y

9. The Belgian mathematician Pierre-François Verhulst designated the sigmoid function the 'logistic curve' in the 1830-40s (Cramer 2004, 616). It was independently designated the 'autocatalytic function' by the German chemist Wilhelm Ostwald in the 1880s, and then re-invented under various names by biologists, physiologists and demographers during 1900-1930s (617). The term 'logistic' returns to visibility in the 1920s, and has continued in use as a way of describing the growth of something that reaches a limit.

function!parameters of

function!referentiality!seealso referentiality

values), but differ in their curvature. This diagrammtic variation derives from the parameter k, which discreetly appears in the equation 3.1 next to x. Such parameters are vital control points in function fitting and any learning associated with that. Varying these parameters and optimising their values is the basis of 'useful approximation' in machine learning. Sometimes these parameters can be varied so much as to suggest entirely different functions. In 3.3 for instance, $k = 12$ produces a much sharper curve, a curve that actually looks more like a qualitative change, range than a smooth transition from 0 to 1. The sharp shape of the logistic function when the scaling parameter k is larger suggests another important transformation, somewhat orthogonal to the description of rates of growth under limits. The mathematical function $f(x) = 1/(1 + e^{-x})$ can be treated as a way of mapping continuously varying numbers (the x values) and discrete values. Because $f(x)$ tends very quickly to converge on values of 1 or 0, it can be coded as 'yes'/'no'; 'survived/deceased', or any other binary difference. The transformation between the x values sliding continuously and the binary difference, classification or categorisation pivots on the combination of the exponential function (e^{-x}), which rapidly tends towards zero as x increases and rapidly tends towards $\inf$ as x decreases, and the \$1/(1+ ...) \$, which converts high value denominators to almost zero, and low value demominators to one. This constrained path between variations in x and their mapping to the value of the function $f(x)$ is mathematically elementary, but typical of the relaying of references that allows functions to intersect with and constitute matters of fact and states of affairs. This realisation – that a continuously varying sigmoid function could also map discrete outcomes – forms the basis of many machine learning classifiers. So, a contemporary biostatistical machine learning textbook can write: we can use logistic regression to 'estimate the probability that a critically-ill lupus patient will not survive the first 72 hours of an initial emergency hospital visit' (Malley, Malley, and Pajevic 2011, 5). If the same curve – the logistic curve – can describe quite different situations (the growth of a population, the probability that someone will die), then we can begin to see that sigmoid functions, and the logistic curve in particular, might be useful devices as approximations to the 'underlying function that generated the data.

function!logistic

machine learner!logistic regression

The cost of curves in machine learning

```
## Error in order(logdf$PY): argument 1 is not a vector
```

How does this take place practically? As I have already mentioned, the logistic function appears frequently in machine learning literature, prominently as part of perhaps the most classical learning machine, the logistic regression model, but also as a component in other techniques such as neural networks (see table 3.2 for a sample of well-cited publications). Descriptions of logistic regression models appear in nearly all machine learning tutorials, textbooks and training courses (see Chapter 4 in (Hastie, Tibshirani, and Friedman 2009)). Logistic regression models are heavily used in biomedical research, where, as 'logistic regression is the default "simple" model for predicting a subject's group status' (Malley, Malley, and Pajevic 2011, 43). As Malley et.al. suggest, 'it can be applied after a more complex learning machine has done the heavy lifting of identifying an important set of predictors given a very large list of candidate predictors' (43). Especially in comparison to more complicated models, logistic regression models are relatively easy to interpret because they are superimposed on the linear model that we have been discussing already (see figure 3.2a and also chapters 2 and **??**). As Hastie et.al write: 'the logistic regression model arises from the desire to model the posterior probabilities of the K classes via linear functions in

x, while at the same time ensuring that they sum to one and remain in $[0,1]$' (Hastie, Tibshirani, and Friedman 2009, 119). Paraphrased somewhat loosely, this says that the logistic regression model predicts what class or category a particular instance is likely to belong to, but 'via linear functions in x.' We see something of this predictive desire from the basic mathematical expression for logistic regression in a situation where there are binary responses or $K = 2$:

$$Pr(G = K|X = x) = \frac{1}{1 + \sum_{l=1}^{K-1} exp(\beta_{l0} + \beta_l^T x)} \tag{3.2}$$

(119)

In equation 3.2, the logistic function operates on linear functions and thus encapsulates lines in curves. That is, the linear model (the model that fits a line to a scattering of points in vector space) appears as $\beta_l 0 + \beta_l^T x$ (where as usual β refers to the parameters of the model and x to the matrix of input values). The linear model has, however, now been put inside the sigmoid function so that its output values no longer increase and decrease linearly. Instead they follow the sigmoid curve of the logistic function, and range between a minimum of 0 and a maximum of 1. As usual, small typographic conventions express some of this shift. In equation 3.2, some new characters appears: G and K. Previously, the response variable, the variable the model is trying to predict, appeared as Y. Y refers to a continuous value whereas G refers to membership of a category or class (e.g. survival vs. death; male vs female; etc.).

What does this wrapping of the linear model in the curve of the sigmoid logistic curve do in terms of finding a function? Note that the shape of this curve has no intrinsic connection or origin in the data. The curve no longer corresponds to growth or change in size, as it did in its nineteenth century biopolitical application to the growth of populations. Rather, the curvilinear encapsulation of the linear model allows the left hand side of the expression to move into a different register. The left hand side of the expression is now a probability function, and defines the probability (Pr) that a given response value (G) belongs to one of the pre-defined classes ($k = 1, ..., K-1$)[10]. In this case, there are two classes ('yes/no'), so $K = 2$. Unlike linear models, that predict continuous y values for a given set of x inputs, the logistic regression model produces a probability that the instance represented by a given set of x values belongs to a particular class. When logistic regression is used for classification, values greater than 0.5 are usually read as class predictions of 'yes', 'true' or `1`. As a result, drawing lines through the common vector space can effectively become a way of classifying things. Note that this increase in flexibility comes at the cost of a loss of direct connection between the data or features in the generalized vector space, and the output, response or predicted variables. They are now connected by a mapping that passes through the somewhat more mobile and dynamic operation of exponentiation exp, a function whose rapid changes can be mapped onto classes and categories.

The cost of curves in machine learning

Whether or not the logistic function is a useful approximation to 'the function that underlies the predictive relationship between input and out,' its combines the features of the common vector space with a reference to a categorical state of affairs in the world. (Other techniques – neural networks or support vector machines – construct different kinds of reference.) The way in which we have 'learned' the logistic function by taking a textbook

10. I leave aside any further discussion of probability in machine learning here. It is the topic of chapter **??**.

dataset!South African heart disease

formula expression of it, and plotting the function associated with it is not the way that machine learners typically 'learns' an approximation to the predictive relationship between the input data and the output variables (the so-called 'response variable'). Finding a function in practice means optimising parameters on the basis of the data. This is not a matter of mathematical analysis, but of algorithmic repetition. [11]

If we turn just to the diagrammatic forms associated with logistic regression in *Elements of Statistical Learning*, something quite different and much more complicated than calculating the values of a known function is going on. For instance, in their analysis of the South African coronary heart disease data, Hastie and co-authors repeatedly model the risk of occurrence of heart disease using logistic regression. They first apply logistic regression fitted by 'maximum likelihood', and then by 'L1 regularized logistic regression' (Hastie, Tibshirani, and Friedman 2009, 126). The results of this function-finding work appear as tables of coefficients or as 'regularization plots.' As is often the case in *Elements of Statistical Learning*, it is assumed that readers already understand conventional statistical usages of logistic regression.

11. Even in machine learning, some function-finding through solving systems of equations occurs. For instance, the closed form solution of the least sum of squares problem for linear regression is given by $\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$. As we saw in the previous chapter, this expression provides a very quick way to calculate the parameters of a linear model given a matrix of input and output values. This formula itself is derived by solving a set of equations for the values $\hat{\beta}$, the estimated parameters of the model. But how do we know whether a model is a good one, or that the function that a model proffers to us fits the functions in our data, or that it 'minimises the probability of error'? One problem with closed-form or analytical solutions typical of mathematical problem-solving is precisely that their closed-form obscures the algorithmic processes needed to actually compute results. The closed form solution estimates the parameters of the linear model by carrying out a series of operations on matrices of the data. These operations include matrix transpose, several matrix multiplications (so-called 'inner product') and matrix inversion (the process of finding a matrix that when multiplied by the input matrix yields the identity matrix, a matrix with 1 along the diagonal, and 0 for all other values). All of these operations take place in the common vector space. When the dataset, however, has a hundred or a thousand rows, these operations can be implemented and executed easily. But as soon as datasets become much larger, it is not easy to actually carry out these matrix operations, particularly the matrix inversion, even on fast computers. For instance, a dataset with a million rows and several dozen columns is hardly unusual today. Although linear algebra libraries are carefully crafted and tested for speed and efficiency, closed form solutions, even for the simplest possible structures in the data, begin to break down.

figure/sa_heart_model-2.pdf

Figure 3.4: South African Heart disease regularization plot

Discussion dwells instead on different ways of fitting models or finding the values of parameters that control the operation of the function.

```
## Error in library(glmnet): there is no package called 'glmnet'

## Error in eval(expr, envir, enclos): could not find function "glmnet"

## Error in plot(SAheart_model2): object 'SAheart_model2' not found

## Error in summary(SAheart_model2): error in evaluating the argument 'object' in selecting

## Error in print(tab2): object 'tab2' not found
```

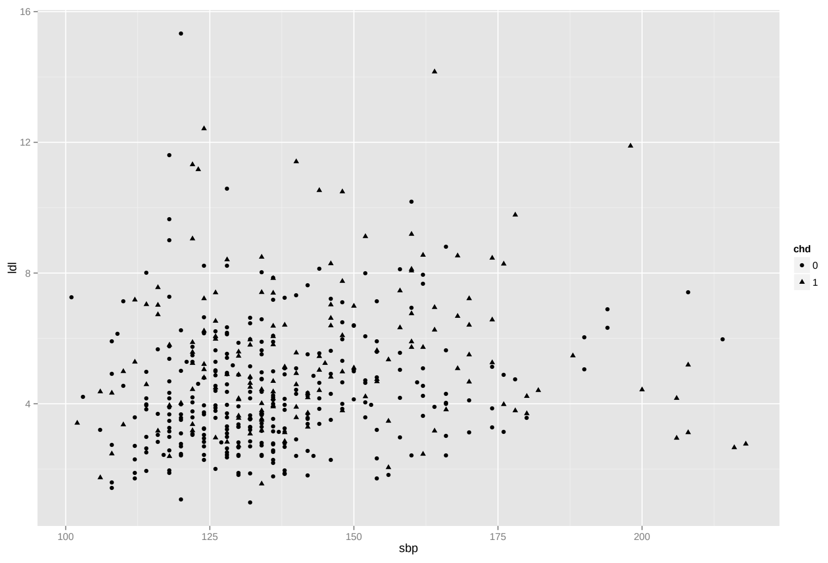


Figure 3.5: South African Heart disease decision plane

Figure 3.4 shows a series of lines. Each line shows the changing importance of a particular variable – obesity, alcohol consumption, weight, age, designated by numbers shown on the right hand side – as it is included in the logistic regression model in a different way. The use of the logistic function does not move through data radically differently. The same sigmoid curve has been used to model binary responses for more than half a century. The learning or function finding diagrammed in figure 3.4 concerns variations in parameters and ways of automating the variation of parameters beyond that undertaken by modelling experts such as statisticians and scientists when they fit models to data. [12]

12. As we saw in chapter **??**, the production of new tables of numbers that list the parameters of models that superimpose different dimensions of the data is a core operation of machine learning. Many of the plots and tables found in machine learning texts, practice and code offer nothing else but measurements of how the model parameters weight different dimensions of the vector space. In the case of logistic regression, the shape of the curve is determined using maximum likelihood. For present purposes, the statistical significance of this procedure is less important than the algorithmic implementation. This is the opposite to what might appear in a typical statistics textbook where the implementation of maximum likelihood would normally be quickly passed over. For instance, in *An Introduction to Statistical Learning with R*, a textbook focused on using `R` to implement machine learning techniques, the authors write: ‘we do not need to concern ourselves with the details of the maximum likelihood fitting procedure’ (James et al. 2013, 133).

The diagram of the changing parameters in a logistic regression model (Figure 3.4 suggests a constrained yet dynamic process of function finding. Machine learners re-shape the parameters of the function within a set of constraints or limits. We saw that the classic statistical model of linear regression fits lines to the data through the method of ordinary least squares (see chapter **??**, equation **??**). The combination of all the variables can be imagined as a surface whose contours include peaks and valleys in common vector space. Many different planes more or less closely fit the contours, and there may be no obvious best one. As we have seen, a linear model tries to find a line or plane or hyperplane (a higher dimensional plane) that aligns with this topography as closely as possible. In mathematical practice more generally, problems are posed and often solved in terms of finding functions. Mathematics textbooks are replete with demonstrations of this problem-solving activity, and mathematics is in large part practiced in solving problems by finding either values or functions that solve problems. These solutions are closed form or analytical solutions.[13]

Several obstacles hinder the construction of models using closed form approximate solutions. Unique 'closed form' or analytical solutions are quite unusual in machine learning. While they do exist for linear regression, they

13. As soon as we move from the more theoretical or expository accounts of function-finding into the domain of practice, instruction and learning of machine learning, a second sense of function comes to the fore. The second sense of function comes from programming and computer science. A function there is a part of the code of a program that performs some operation, 'a self-contained unit of code,' as Derek Robinson puts it (Robinson 2008, 101). The three lines of R code written to produce the plot of the logistic function are almost too trivial to implement as a function in this sense, but they show something of the transformations that occur when mathematical functions are operationalised in algorithmic form. The function is wrapped in a set of references. First, the domain of x values is made much more specific. The formulaic expression $f(x) = 1/(1+e^{-x})$ says nothing explicitly about the x values. They are implicitly real numbers (that is, $x \in \mathbb{R}$) in this formula but in the algorithmic expression of the function they become a sequence of 20001 generated by the code. Second, the function itself is flattened into a single line of characters in code, whereas the typographically the mathematical formula had spanned 2-3 lines. Third, a key component of the function $e^{-}x$ itself refers to Euler's number e, which is perhaps the number most widely used in contemporary sciences due to its connection to patterns of growth and decay (as in the exponential function e^x where $e = 2.718282$ approximately). This number, because it is 'irrational,' has to be computed approximately in any algorithmic implementation.

machine learner!optimisation of

don't exist for logistic regression nor for complex machine learners. Equally problematically, the closed form solution is run once, and the model it produces is subject to no further variation. The parameters define the line of best fit. It can be interpreted by the modeller in terms of p or R^2 or other measures of the model's fit. But the model itself does not generate variations.

Cost, loss, objective and optimisation

Instead, machine learners typically seek ways of observing how different models traverse the data. They replace the exactitude and precision of mathematically-deduced closed-form solutions with algorithms that generate a variety of solutions. A range of optimisation techniques search for optimal combinations of parameters. These optimisation techniques are the operational underpinning of machine learning. Without their iterative processes, their is no machine in machine learning. They have names such as 'batch gradient descent', 'stochastic gradient ascent,' 'coordinate descent,' 'coordinate ascent' as well as the 'Newtown-Raphson method' or simply 'convex optimisation' (Boyd and Vandenberghe 2004). These techniques have a variety of provenances (Newton's work in the 17th century, for instance, but more typically fields such as operations research that were the focus of intense research efforts during and after WWII; see (Bellman 1961; Petrova and Solov'ev 1997; Meza 2010)). Much of the learning in machine learning occurs through these somewhat low-profile yet computationally intensive numerical techniques.

Optimisation is a practice of reference. 'Science brings to light partial observers in relation to functions within systems of reference' write Gilles Deleuze and Félix Guattari in their account of scientific functions (Deleuze and Guattari 1994, 129).[14] In many machine learning techniques, for instance, the search for an optimised approximation to the function that generated the data is guided by another function called the 'cost function' (also known as the 'objective function' or the 'loss function'; all the terms are somewhat evocative). Typically, machine learning problems are framed in terms of minimizing or maximising the cost function. The value of the cost function is usually understood in terms of errors, and minimizing the cost function implies minimizing the number of errors made by a machine learner. As we saw earlier, in his formulation of the 'learning problem', the learning theorist Vladimir Vapnik speaks of choosing a function that approximates to the data, yet minimises the 'probability of error' (Vapnik 1999, 31). Even the apparently simplest data modelling procedure of fitting a line to a set of points is usually implemented as an optimisation process in machine learning settings.

function!as partial observer

function!cost or objective

The cost function measures how predictions generated by a machine learner compare to known values in the data set. Every cost function implies some measure of the difference or distance between the prediction and the values actually measured. In classifying outcomes into two classes (the patient survives versus patient dies; the user clicks versus user doesn't click; etc.), the cost function has to express their either/or outcome. Crucially, if cost functions re-configure 'the act of fitting a model to data as an optimization problem' (Conway and White 2012, 183), function finding occurs iteratively not analytically. Given a cost function, a machine learner can shape its variation of models parameters keeping in view – or partially observing – whether the cost function increases or decreases. The cost functions add a diagrammatic layer in which the relation between models, and particularly

14. Its hard to know whether Deleuze and Guattari were aware of the extensive work done on problems of mathematical optimization during the 1950-1960s, but their strong interest in the differential calculus as a way of thinking about change, variation and multiplicities somewhat unexpectedly makes their account of functions highly relevant to machine learning.

machine learner!learning of

their predictive reference becomes visible. If there is learning here, it is not some enigmatic form or a higher form of scientific reason. Just the opposite, the cost function does something more like create a place from which variations in models can be viewed. A typical and widely-used cost function associated with logistic regression is defined as:

$$J(\beta) = \sum_{i=1}^{m} y_i logh(x_i) + (1 - y_i) log(1 - h(x_i)) \tag{3.3}$$

where

$$h_\beta(x) = 1/1 + e^{-\beta^T x}$$

Equation 3.3 enfolds several manipulations and conceptual framings (particularly the Principle of Maximum Likelihood, a statistical principle; see chapter **??**. But key terms stand out. First, the cost function $J(\beta)$ is a function of all the parameters (β) of the model. Second, the function defines a goal of maximising the overall value of the expression. The min describes the results of the repeated application of the function. Third, the heart of the function is a kind of average: it adds ($\sum$) all the values where the probability of the predicted class of a particular case $h(x_i)$ matches the actual class, and subtracts $(1 - y)$ all the cases where the probability of the predicted class does not match the actual class. This so-called *log likelihood* function can be maximised, but not solved in closed form. The optimal values for β, the model parameters that define the model function need to be found through some kind of search.

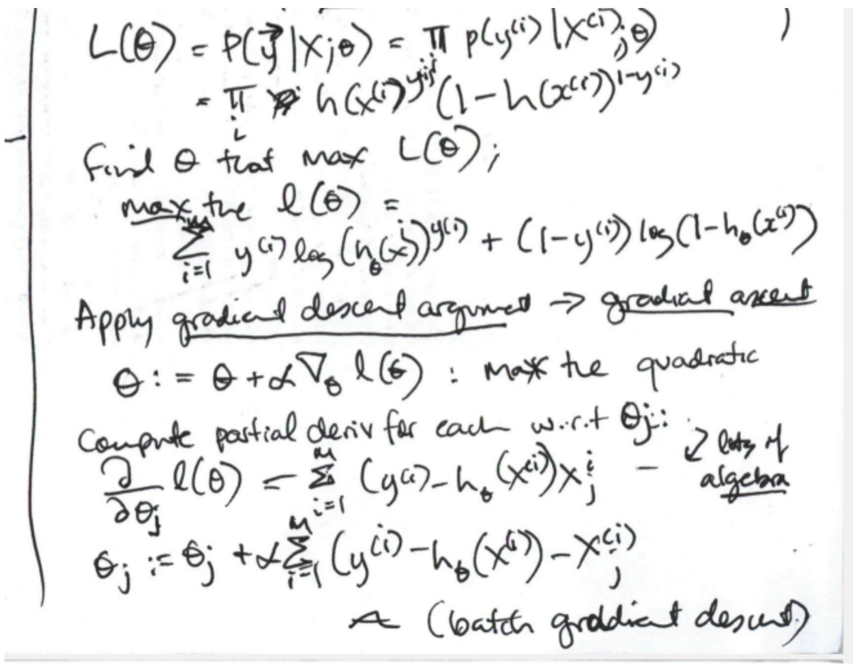
Gradients as partial observers

Many of the optimization techniques used in machine learning rely on differential calculus. One widely used optimisation algorithm called 'gradient descent' is quite easy to grasp intuitively. As in many formulations of machine learning techniques, the framing of the problem is finding the parameters of the model/function that best approximates to the function that generated the data. It optimises the parameters of a model by searching for the maximum or minimum values of the objective function. The algorithm can be written using calculus style notation as:

Repeat until convergence:

$$\beta_j := \beta_j + \alpha(y_i - h_\beta(x_i))x_{\beta j} \tag{3.4}$$

The version of the algorithm shown in algorithm **??** is called 'stochastic gradient descent.' As always, in presenting these diagrammatic formula, the point is not to read and understand them directly. (That would be the point in a machine learning course.) Actually reading these formal expressions, and being able to follow the chain of references, and indexical signs that lead away from them in various directions depends very much on the diagrammatic processes described in Chapter 1. Many people who directly use machine learning techniques in industry and science would not often if ever need to make use of such expressions as they build models. They would mostly take them for granted, and simply execute them. My purpose here, however, is a bit different. Rather than explaining these formulations, my interest is following the threads and systems of reference that wind through them, and to identify the points of transition, friction or slippage that both allow them to work and also not be everything they



$L(\theta) = P(\vec{y} \mid X; \theta) = \prod p(y^{(i)} \mid x^{(i)}; \theta)$

$= \prod_i h(x^{(i)})^{y^{(i)}} (1 - h(x^{(i)}))^{1-y^{(i)}}$

Find θ that max $L(\theta)$;

max the $\ell(\theta) =$

$\sum_{i=1} y^{(i)} \log (h_\theta(x^{(i)}))^{y^{(i)}} + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))$

Apply gradient descent argument $\rightarrow$ gradient ascent

$\theta := \theta + \alpha \nabla_\theta \ell(\theta)$: max the quadratic

Compute partial deriv for each w.r.t θ_j:

$\frac{\partial}{\partial \theta_j} \ell(\theta) = \sum_{i=1}^{m} (y^{(i)} - h_\theta(x^{(i)})) x_j^{(i)}$ — lots of algebra

$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} (y^{(i)} - h_\theta(x^{(i)})) - x_j^{(i)}$

(batch gradient descent)

Figure 3.6: Gradient ascent for logistic regression

claim to be.

Given that this expression encapsulates the heart of a major optimisation technique, we might first of all be struck by its operational brevity. This is not an elaborate or convoluted algorithm. As Malley, Mally and Pajevic observe, 'most of the [machine learning] procedures … are (often) nearly trivial to implement' (Malley, Malley, and Pajevic 2011, 6). Note that this expression of the algorithm, taken from the class notes for [Lecture 3] of Andrew Ng's 'Machine Learning' CS229 course at Stanford (see Figure 3.6), mixes an algorithmic set of operations with function notation. We see this in several respects: the formulation includes the imperative 'repeat until convergence'; it also uses the so-called 'assignment operator' := rather than the equality operator =. The latter specifies that two values or expressions are equal, whereas the former specifies that the values on the right hand side of the expression should be assigned to the left. Both algorithmic forms – repeat until convergence, and assign/update values – owe more to techniques of computation than to mathematical abstraction. In this respect more generally, by looking at implementations we begin to see how systems of references are put together. In this case, the specification for the gradient descent algorithm starts to bring us to the scene where data is actually reshaped according by the more abstract mathematical functions we have been describing (mainly using the example of the linear regression and its classic algebraic form $y = \beta_0 + \beta_1 x_1 + ...\beta_n x_n$). At the heart of this reshaping lies a different mathematical formalism: the partial derivative, $\frac{\partial}{\partial \beta_j} J(\beta_j)$. Like all derivatives in calculus, this expression can be interpreted as a rate; that is as the rate at which the cost function $J(\beta)$ changes with respect to the different values of β [15]. If there is any learning in machine learning, it pivots on the observation of movement. The partial derivatives in the gradient descent algorithm observe the direction in which the cost function becomes smaller. On each iteration of the algorithm, the parameters β of the model move in that direction of reduced cost, and perhaps less error.

function!diagrammatic operation of

diagram!mathematical function as

The power of machine learning to learn, then, pivots around functions in disparate ways: the diagramming of functions in mathematical expressions, in the graphic forms of plots of lines and curves of the changing shape of functions, and in the algorithms that superimpose new functions – cost, loss or objective functions – in an iterative process of observation and modification. Machine learning diagrammatically distributes learning between people and things. People looking at curves, functions compressing data into parameters that support classification or predictions, and algorithms observing gradients. In several senses, people and machines together move along curves. The logistic function folds the lines that best fit the data into a probability distribution that can be read in terms of classification. The loss functions seek convergence between the predicted values and the known

15. The derivative $\frac{\partial}{\partial \beta_j} J(\beta_j)$ is *partial* because β is a vector $\beta_0, \beta_1 ... \beta_j$.

diagram|diagonal

function!as partial observer

values found in the common vector space. The gradient-based algorithms look for the direction in which the peaks or valleys lie. Everywhere the diagram switches and diagonalizes between the human observers and machine observers. Every observer in this domain is partial, since the humans cannot see lines or curves in the multi-dimensional data, the functions that underpin models such as logistic regression or linear regression can traverse data in the common vector space, but can't show how well they see it, and the processes of optimisation only see the results of the model and its errors, not anything in its referential functioning. Impartiality, whether fully supervised or completely unsupervised, is impossible here.

Amidst this chronic partiality, we can begin to understand the multiplication of functions and machine learners. Machine learners are functions that classify, predict and rank, but the function that defines a 'machine learner' contracts a range of partial observers opaquely associated with each other. If, as I have been suggesting, the algorithmic power of machine learning derives from functions (and I think in a literal technical sense, that is what much machine learning literature states at length in great detail), its operational and referential power depends on the diagrammatic and sometimes experimental relations between different practices of observing. Attending to specific mathematical functions in isolation – the logistic function, the Lagrangean, the Gaussian, the quadratic discriminant, etc. – will not tell us how the operational power of functions comes together in machine learning, but it may provide ways of mapping the diagrammatic connections that generate reference operational and referential power. In his account of diagrams of power, Gilles Deleuze writes:

> every diagram is intersocial and constantly evolving. It never functions in order to represent a persisting world but produces a new kind of reality, a new model of truth. ... It makes history

> by unmaking preceding realities and significations, constituting hundreds of points of emergence or creativity, unexpected conjunctions or improbable continuums (Deleuze 1988, 35)

diagram!of power

Deleuze, Gilles!seealsodiagram!of power

This somewhat theoretical description of diagrams might help make sense of the learning in machine learning. Functions in machine learning are 'intersocial' in the sense that they bring together very different mathematical, algorithmic, operational and referential processes. The sigmoid function switches the geometry of the linear model over into the calculation of probabilities and classification. The cost functions re-craft statistical modelling as a quasi-iterative process of model generation and comparison. New kinds of realities arise in which the classifications and predictions generated by the diagonal connections between mathematical functions and operational processes of optimisation can constitute a 'new model truth' and can unmake 'preceding realities and significations.' And despite my deliberately narrow focus on a single set of diagonals that connects linear models, the logistic function, the cost function and gradient ascent, there are hundreds and perhaps and hundreds of thousands of 'points of emergence' associated with this diagram of functioning. They are scattered across sciences, industry, government and commerce, and appear in manifold variations.

Agency dances along the curves. Curves traverse and encapsulate many lines. They become mobile partial observers of many lines. Animating the curves, and then looking at those optimising animations as 'learning' generates operational power dynamics. The diagram attracts infrastructural, technical, professional, semiotic and financial diagonals that render its traits more real, more thickly transformative and, it goes almost without saying, more performant. In the history of automata, automation and animation, kinetic lures have long exercised fascination, and this may be part of the effect of machine learning. Yet such performant diagrams generate referential effects.

Lazzarato, Maurizio!asemiotic machine

Machine learning becomes ontologically potent. As Maurizio Lazzarato writes in *Signs and Machines*, 'ontological mutations are always machinic. They are never the simple result of the actions or choices of the "man" who, leaving the assemblage, removes himself from the non-human, technical, or incorporeal elements that constitute him – all that is pure abstraction' (Lazzarato 2014, 83). The differences of function held together in machine learners (bearing in mind the double meaning of that term) hold together diagrammatically, but generate statements and visual objects that count as knowledge, truth and increasingly as action.

The machine learning diagram, like any semiotic system, harbours the potential for invention. For instance, describing the application of machine learning to biomedical and clinical research, James Malley, Karen Malley and Sinisa Pajevic contrast it to more conventional statistical approaches:

> working with statistical learning machines can push us to think about novel structures and functions in our data. This awareness is often counterintuitive, and familiar methods such as simple correlations, or slightly more evolved partial correlations, are often not sufficient to pin down these deeper connections. (Malley, Malley, and Pajevic 2011, 5-6)

The novel structures and functions in 'our data' are precisely the functions that machine learning technique seek to learn. These structures and functions take various forms and shapes (lines, trees, curves, peaks, valleys, forests, boundaries, neighbourhoods, etc.), and they can identify 'deeper connections' than correlations. This is where new habits and 'actively diverging worlds' (Stengers) might appear. But note that the function itself in isolation never learns. It is always watched or observed in some way, even just virtually. Once the observation stops (for instance, when the machine learner has become part of a device such as a voice recognition system or spam classifier), then the inter-social evolution of the diagram pauses, and it persists in recognising a pattern.

Year	Title	Keywords	Citations
1995	Recurrent Radial Basis Function Networks For Adaptive Noise Cancellation	radial basis function network; noise cancellation; nonlinear filtering	65
1997	Comparing Support Vector Machines With Gaussian Kernels To Radial Basis Function Classifiers	clustering; pattern recognition; prototypes; radial basis function networks; support vector machines	392
1998	Application Of Radial Basis Function And Feedforward Artificial Neural Networks To The Escherichia Coli Fermentation Process	radial basis function network; feedforward neural network; bioprocess monitoring; biomass estimation; recombinant process modeling	31
1998	Rbf Nets, Mixture Experts, And Bayesian Ying Yang Learning	radial basis function network; mixture of experts; bayesian ying-yang learning; model selection; coordinated competitive learning; fast computation; adaptive algorithm; financial time series; curve fitting	67
1999	A Kurtosis Based Dynamic Approach To Gaussian Mixture Modeling	expectation-maximization (em) algorithm; gaussian mixture modeling; number of mixing kernels; probability density function estimation; total kurtosis; weighted kurtosis	40
2001	Greedy Function Approximation: A Gradient Boosting Machine	function estimation; boosting; decision trees; robust nonparametric regression	900
2002	Technical Update: Least Squares Temporal Difference Learning	reinforcement learning; temporal difference learning; value function approximation; linear least-squares methods	50
2003	Acoustic Identification Of Nine Delphinid Species In The Eastern Tropical Pacific Ocean	species identification; towed hydrophone array; sonobuoy; discriminant function analysis; decision tree; dolphin; whistle; acoustic; stenella longirostris; stenella attenuata; stenella coeruleoalba; delphinus delphis; delphinus capensis; tursiops truncatus; steno bredanensis; globicephala macrorbynchus; pseudorca crassidens	42
2004	Discriminative Learning Quadratic Discriminant Function For Handwriting Recognition	discriminative learning quadratic discriminant function (dlqdf); handwritten digit recognition; minimum classification error (mce); noncharacter resistance; numeral string recognition; pattern classification	46
2004	A Function Decomposition Method For Development Of Hierarchical Multi Attribute Decision Models	multi-attribute decision making; hierarchical models; function decomposition; data-driven modeling; data mining	23
2005	Smooth Function Approximation Using Neural Networks	algebraic; function approximation; gradient; input-output; training	81
2006	Soft Sensor Development For Fed Batch Bioreactors Using Support Vector Regression	artificial neural networks; bioreactor; soft-sensors; support vector regression; multilayer perceptron; radial basis function network	41
2006	A Wavelet Network Model For Short Term Traffic Volume Forecasting	wavelet networks; back-propagation neural networks; radial basis function neural networks; levenberg-marquardt algorithm; traffic volume forecasting	30
2006	Improving Rbf Networks Performance In Regression Tasks By Means Of A Supervised Fuzzy Clustering	fuzzy clustering; fuzzy c-means; radial basis function neural networks; linear regression models	32
2006	An Explicit Description Of The Reproducing Kernel Hilbert Spaces Of Gaussian Rbf Kernels	gaussian radial basis function (rbf) kernel; reproducing kernel hilbert space; support vector machine	28
2006	Structural Bioinformatics Prediction Of Membrane Binding Proteins	protein-membrane interactions; function annotation; support vector machines; peripheral proteins; protein function prediction	34
2007	Choosing Parameters Of Kernel Subspace Lda For Recognition Of Face Images Under Pose And Illumination Variations	gaussian radial basis function (rbf) kernel; generalization capability; kernel fisher discriminant (kfd); kernel parameter; model selection	33
2008	A Sequential Multi Category Classifier Using Radial Basis Function Networks	rbf network; hinge loss function; sequential learning; multi-category classification; decoupled extended kalman filter	33
2009	Dictionary Learning For Sparse Approximations With The Majorization Method	block relaxation methods; constrained optimization; dictionary learning; majorization methods; sparse approximation; surrogate function optimization method	27
2009	Computational Chemistry Study Of 3d Structure Function Relationships For Enzymes Based On Markov Models For Protein Electrostatic, Hint, And Van Der Waals Potentials	3d-qsar; markov chains; protein structure-function relationship; electrostatic potential; van der waals potential; hint potential; enzymes; machine learning; artificial neural networks	26
2009	Time Series Prediction Using Rbf Neural Networks With A Nonlinear Time Varying Evolution Pso Algorithm	time series prediction; radial basis function networks; particle swarm optimization; nonlinear time-varying evolution	49
2010	Mapping Mental Function To Brain Structure: How Can	ontology; neuroimaging; machine	24

Year	Title	Keywords	Citations
2010	Regularization Paths For Generalized Linear Models Via Coordinate Descent	Lasso; Elastic Net; Logistic Regression; L(1) Penalty; Regularization Path; Coordinate Descent	912
2010	Regularization Paths For Generalized Linear Models Via Coordinate Descent	Lasso; Elastic Net; Logistic Regression; L(1) Penalty; Regularization Path; Coordinate Descent	859
2002	Evaluating Resource Selection Functions	Habitat Selection; Logistic Regression; Model Selection; Prediction; Rsf; Resource Selection Functions; Validation	587
2010	Regularization Paths For Generalized Linear Models Via Coordinate Descent	Lasso; Elastic Net; Logistic Regression; L(1) Penalty; Regularization Path; Coordinate Descent	565
2007	Random Forests For Classification In Ecology	Additive Logistic Regression; Classification Trees; Lda; Logistic Regression; Machine Learning; Partial Dependence Plots; Random Forests; Species Distribution Models	563
1995	Bayesian Computation And Stochastic Systems	Agricultural Field Experiments; Bayesian Inference; Conditional Distributions; Deconvolution; Gamma Camera Imaging; Gibbs Sampler; Hastings Algorithms; Image Analysis; Logistic Regression; Markov Chain Monte Carlo; Markov Random Fields; Metropolis Method; Prostate Cancer; Simultaneous Credible Regions; Spatial Statistics; Time Reversibility; Unobserved Covariates; Variety Trials	495
2007	Random Forests For Classification In Ecology	Additive Logistic Regression; Classification Trees; Lda; Logistic Regression; Machine Learning; Partial Dependence Plots; Random Forests; Species Distribution Models	427
2008	Liblinear: A Library For Large Linear Classification	Large Scale Linear Classification; Logistic Regression; Support Vector Machines; Open Source; Machine Learning	316
1996	Advantages And Disadvantages Of Using Artificial Neural Networks Versus Logistic Regression For Predicting Medical Outcomes	Neural Networks; Logistic Regression	270
2008	The Group Lasso For Logistic Regression	Categorical Data; Co Ordinate Descent Algorithm; Dna Splice Site; Group Variable Selection; High Dimensional Generalized Linear Model; Penalized Likelihood	266
2005	Sparse Multinomial Logistic Regression: Fast Algorithms And Generalization Bounds	Supervised Learning; Classification; Sparsity; Bayesian Inference; Multinomial Logistic Regression; Bound Optimization; Expectation Maximization (em); Learning Theory; Generalization Bounds	260
2007	Random Forests For Classification In Ecology	Additive Logistic Regression; Classification Trees; Lda; Logistic Regression; Machine Learning; Partial Dependence Plots; Random Forests; Species Distribution Models	258
1999	Comparing Discriminant Analysis, Neural Networks And Logistic Regression For Predicting Species Distributions: A Case Study With A Himalayan River Bird	Neural Networks; Logistic Regression; Presence Absence Data; River Birds	224
2001	Prognostic Modeling With Logistic Regression Analysis: In Search Of A Sensible Strategy In Small Data Sets	Regression Analysis; Logistic Models; Bias; Variable Selection; Prediction	208
2002	New Strategies For Identifying Gene Gene Interactions In Hypertension	Data Reduction; Epistasis; Essential Hypertension; Gene Gene Interaction; Hardy Weinberg Disequilibrium; Linkage Disequilibrium; Logistic Regression; Multifactor Dimensionality Reduction; Pattern Recognition	200
2006	Resource Selection Functions Based On Use Availability Data: Theoretical Motivation And Evaluation Methods	Bias; Contaminated Control; Habitat Modeling; Logistic Discriminate; Logistic Regression; Resource Selection Function; Rsf; Sampling; Design; Use Availability; Validation	192
2009	Genome Wide Association Analysis By Lasso Penalized Logistic Regression	NANA	191
2006	Modelling Distribution And Abundance With Presence Only Data	Case Control; Distribution; Habitats; Logistic Discrimination; Logistic Regression; Presence Only Studies; Pseudo Absences; Resource Selection Functions; Rsf; Sampling	191
2002	Logistic Regression And Artificial Neural Network Classification Models: A Methodology Review	Artificial Neural Networks; Logistic Regression; Classification; Model Comparison; Model Evaluation; Medical Data Analysis	184
2005	Sparse Multinomial Logistic Regression: Fast Algorithms And Generalization Bounds	Supervised Learning; Classification; Sparsity; Bayesian Inference; Multinomial Logistic Regression; Bound Optimization; Expectation Maximization (em); Learning Theory; Generalization Bounds	176
1999	Stepwise Selection In Small Data Sets: A Simulation	Regression Analysis; Logistic	163

Glossary

classifier A machine learner that assigns instances to classes or categories..

6

Bibliography

Abramowitz, Milton. 1965. *Handbook of mathematical functions : with formulas,graphs,and mathematical tables.* In collaboration with Irene A. Stegun and United States. National Bureau of Standards. Dover Books.

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.

Agency, National Security. 2012. “SKYNET: Courier Detection via Machine Learning.” Accessed October 29, 2015. https://theintercept.com/document/2015/05/08/skynet-courier/.

Alpaydin, Ethem. 2010. *Introduction to machine learning.* Cambridge, Massachusetts; London: MIT Press.

Amoore, Louise. 2011. “Data Derivatives On the Emergence of a Security Risk Calculus for Our Times.” *Theory, Culture & Society* 28 (6): 24–43.

Aristotle. 1975. *Aristotle’s Categories and de Interpretatione.* Translated by J.L. Ackrill. Oxford University Press.

———. 1981. *Nicomachean Ethics.* Hammondsworth, UK: Penguin Books.

Arthur, Heather. 2012. “harthur/kittydar.” Accessed September 16, 2014. https://github.com/harthur/kittydar.

Barad, Karen. 2007. *Meeting the universe halfway: Quantum physics and the entanglement of matter and meaning.* Duke University Press Books.

Barber, David. 2011. *Bayesian reasoning and machine learning.* Cambridge; New York: Cambridge University Press.

Barocas, Solon, Sophie Hood, and Malte Ziewitz. 2013. *Governing Algorithms: A Provocation Piece.* SSRN SCHOLARLY PAPER ID 2245322. Rochester, NY: Social Science Research Network.

BBC. 2012. “Google ’brain’ machine spots cats.” *BBC News: Technology* (June 26).

Beer, David, and Roger Burrows. 2013. “Popular culture, digital archives and the new social life of data.” *Theory, Culture & Society.*

Bellman, Richard. 1961. *Adaptive control processes: a guided tour.* Vol. 4. Princeton university press Princeton.

Beniger, James R. 1986. *The control revolution: technological and economic origins of the information society.* Harvard University Press.

Bishop, Christopher M. 2006. *Pattern recognition and machine learning.* Vol. 1. New York: Springer.

Bogost, Ian. 2012. *Alien phenomenology, or what it’s like to be a thing.* U of Minnesota Press.

Bollas, Christopher. 2008. *The evocative object world.* London & New York: Routledge.

Boyd, Stephen, and Lieven Vandenberghe. 2004. *Convex optimization.* Cambridge ; New York: Cambridge university press.

Breiman, Leo. 2001. “Statistical modeling: The two cultures (with comments and a rejoinder by the author).” *Statistical Science* 16 (3): 199–231.

Breiman, Leo, Jerome Friedman, Richard Olshen, Charles Stone, D. Steinberg, and P. Colla. 1984. “CART: Classification and regression trees.” *Wadsworth: Belmont, CA* 156.

Butler, Declan. 2013. “When Google got flu wrong.” *Nature* 494, no. 7436 (February 13): 155–156.

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

Cassirer, Ernst. 1923. *Substance and Function.* Translated by William Curtis Swabey and Marie Curtis Swabey. Chicago: Open Court Publishing.

Cheney-Lippold, John. 2011. “A new algorithmic identity soft biopolitics and the modulation of control.” *Theory, Culture & Society* 28 (6): 164–181.

Cleveland, William S., Eric Grosse, and William M. Shyu. 1992. “Local regression models.” *Statistical models in S:* 309–376.

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.

Couldry, Nick. 2012. *Media, society, world: Social theory and digital media practice.* Cambridge ; Malden, MA: Polity.

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

Cramer, J. S. 2004. "The early origins of the logit model." *Studies in History and Philosophy of Science Part C: Studies in History and Philosophy of Biological and Biomedical Sciences* 35 (4): 613–626.

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.

Deleuze, Gilles, and Félix Guattari. 1994. *What is philosophy?* Translated by Hugh Tomlinson. European perspectives. New York; Chichester: Columbia University Press.

Domingos, Pedro. 2012. "A few useful things to know about machine learning." *Communications of the ACM* 55 (10): 78–87.

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 (February 1): 5–30.

Fico. 2015. "FICO® Analytic Modeler Decision Tree Professional | FICO™." Accessed November 1, 2015. http://www.fico.com/en/products/fico-analytic-modeler-decision-tree-professional.

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.

Frey, Carl Benedikt, and Michael Osborne. 2013. *The Future of Employment: How susceptible are jobs to computerisation?* Oxford: Oxford Martin School, Oxford University.

Fuller, Matthew, and Andrew Goffey. 2012. *Evil Media.* Cambridge, Mass: MIT Press.

Galloway, Alexander. 2014. “The Cybernetic Hypothesis.” *differences* 25, no. 1 (January 1): 107–131.

Galloway, Alexander R. 2004. *Protocol: how control exists after decentralization.* Leonardo (Series) (Cambridge, Mass.) Cambridge, Mass.: MIT Press.

Gillespie, Tarleton. 2010. “The politics of ’platforms’.” *New Media & Society* 12 (3): 347–364.

———. 2014. “9 The Relevance of Algorithms.” *Media technologies: Essays on communication, materiality, and society:* 167.

Gitelman, Lisa, ed. 2013. *”Raw Data” is an Oxymoron.* Cambridge, Massachusetts ; London, England: MIT Press.

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.

Hallinan, Blake, and Ted Striphas. 2014. “Recommended for you: The Netflix Prize and the production of algorithmic culture.” *New Media & Society* (June 23): 1–21.

Hardy, Quentin. 2010. “Power in the Numbers Page 2 of 3 - Forbes.com.” May 24. Accessed June 10, 2010. http://www.forbes.com/forbes/2010/0524/opinions-software-norman-nie-spss-ideas-opinions_2.html.

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.

Heis, Jeremy. 2014. “Ernst Cassirer’s Substanzbegriff und Funktionsbegriff.” *HOPOS: The Journal of the International Society for the History of Philosophy of Science* 4, no. 2 (September 1): 241–270. JSTOR: 10.1086/676959.

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.

Hothorn, Torsten. 2014. “CRAN Task View: Machine Learning & Statistical Learning” (December 18).

IBM. 2013. “IBM PureData System.” Accessed June 21, 2013. http://www-01.ibm.com/software/data/puredata/.

Issenberg, Sasha. 2012. “The Definitive Story of How President Obama Mined Voter Data to Win A Second Term | MIT Technology Review.” Accessed January 9, 2013. http://www.technologyreview.com/featuredstory/509026/how-obamas-team-used-big-data-to-rally-voters/.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An introduction to statistical learning.* Springer.

Jockers, Matthew L. 2013. *Macroanalysis: Digital Methods and Literary History.* Urbana: University of Illinois Press.

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.

Lanier, Jaron. 2013. *Who owns the future?* London: Allen Lane.

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

Larsen, Jeff. 2012. "How ProPublica's Message Machine Reverse Engineers Political Microtargeting." Accessed August 28, 2014. http://www.propublica.org/nerds/item/how-propublicas-message-machine-reverse-engineers-political-microtargeting.

Lash, Scott. 2007. "Power after Hegemony: Cultural Studies in Mutation?" *Theory, Culture & Society* 24 (3): 55–78.

Latour, Bruno. 1993. *We have never been modern.* New York ; London: Harvester Wheatsheaf.

Lazer, David, Alex Sandy Pentland, Lada Adamic, Sinan Aral, Albert Laszlo Barabasi, Devon Brewer, Nicholas Christakis, Noshir Contractor, James Fowler, and Myron Gutmann. 2009. "Life in the network: the coming age of computational social science." *Science (New York, NY)* 323 (5915): 721.

Lazzarato, Maurizio. 2014. *Signs and Machines: Capitalism and the Production of Subjectivity.* Cambridge, MA: Semiotext (e).

Le, Quoc V., Marc'Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg S. Corrado, Jeff Dean, and Andrew Y. Ng. 2011. "Building high-level features using large scale unsupervised learning" (December 28). arXiv: 1112.6209.

Lecture 1 | Machine Learning (Stanford). 2008.

Lee, D. D., and H. S. Seung. 1999. "Learning the parts of objects by non-negative matrix factorization." *Nature* 401, no. 6755 (October 21): 788–791. pmid: 10548103.

Mackenzie, Adrian. 1997. "Undecidability: the history and time of the Universal Turing Machine." *Configurations* 3:359–379.

———. 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.

Mackenzie, Adrian, Matthew Fuller, Andrew Goffey, Mills , Richard, and Stuart Sharples. 2016. "Code repositories as expressions of urban life." In *Code and the City,* edited by Rob Kitchin. London: Routledge.

Madrigal, Alexis C. 2014. "How Netflix Reverse Engineered Hollywood." Accessed August 28, 2014. http://www.theatlantic.com/technology/archive/2014/01/how-netflix-reverse-engineered-hollywood/282679/.

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.

Massumi, Brian. 2013. *Semblance and Event: Activist Philosophy and the Occurrent Arts.* Reprint edition. Cambridge, Mass.: MIT Press, September 6.

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

McCormack, Derek. 2012. "Geography and abstraction Towards an affirmative critique." *Progress in Human Geography* 36, no. 6 (December 1): 715–734.

McKinsey. 2009. "Hal Varian on how the Web challenges managers -." Accessed June 10, 2010. http://www.mckinseyquarterly.com/Hal_Varian_on_how_the_Web_challenges_managers_2286.

McMillan, Robert. 2013. "How Google Retooled Android With Help From Your Brain." February 18. Accessed August 4, 2015. http://www.wired.com/2013/02/android-neural-network/.

Meza, Juan C. 2010. "Steepest descent." *Wiley Interdisciplinary Reviews: Computational Statistics* 2 (6): 719–722.

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.

Mohamed, Abdel-rahman, Tara N. Sainath, George Dahl, Bhuvana Ramabhadran, Geoffrey E. Hinton, and Michael A. Picheny. 2011. "Deep Belief Networks using discriminative features for phone recognition," 5060–5063. IEEE, May.

Mohr, John W., and Petko Bogdanov. 2013. "Introduction—Topic models: What they are and why they matter." *Poetics* 41, no. 6 (December): 545–569.

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

Munster, Anna. 2013. *An Aesthesia of Networks: Conjunctive Experience in Art and Technology.* MIT Press.

Neyland, Daniel. 2014. "On Organizing Algorithms." *Theory, Culture & Society:* 0263276414530477.

Olazaran, Mikel. 1996. "A Sociological Study of the Official History of the Perceptrons Controversy." *Social Studies of Science* 26, no. 3 (January 8): 611–659.

Parisi, Luciana. 2013. *Contagious Architecture: Computation, Aesthetics and Space.* Cambridge ; Malden, MA: MIT Press.

Pasquinelli, Matteo. 2014. "Italian Operaismo and the Information Machine." *Theory, Culture & Society* (February 2): 1–20.

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 (November 1): 559–572.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 2011. “Scikit-learn: Machine Learning in Python.” *Journal of Machine Learning Research* 12:2825–2830.

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.

Petrova, Svetlana S., and Alexander D. Solov’ev. 1997. “The Origin of the Method of Steepest Descent.” *Historia Mathematica* 24, no. 4 (November): 361–375.

R Development Core Team. 2010. “The R Project for Statistical Computing.” Accessed June 11, 2010. http://www.r-project.org/.

RexerAnalytics. 2010. “Rexer Analytics 4th Annual Data Miner Survey - 2010.” Accessed May 9, 2011. http://www.rexeranalytics.com/Data-Miner-Survey-Results-2010.html.

Ripley, Brian. 1996. *Pattern recognition and neural networks. 1996.* Cambridge ; New York: Cambridge University Press.

Robinson, Derek. 2008. "Function." In *Software studies: a lexicon,* edited by M. Fuller, 101–110. The MIT 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.

Smith, David. 2012. "R analysis shows how UK health system could save £200m." Accessed December 12, 2013. http://blog.revolutionanalytics.com/2012/12/nhs-prescription-analytics.html.

Smith, Marquard. 2013. "Theses on the Philosophy of History: The Work of Research in the Age of Digital Searchability and Distributability." *Journal of Visual Culture* 12, no. 3 (December 1): 375–403.

Stengers, Isabelle. 2005. "Deleuze and Guattari's Last Enigmatic Message." *Angelaki* 10 (1): 151–167.

———. 2008. "Experimenting with Refrains: Subjectivity and the Challenge of Escaping Modern Dualism." *Subjectivity* 22, no. 1 (May): 38–59.

———. 2011. *Cosmopolitics II.* Translated by Robert Bononno. University of Minnesota Press, September 26.

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, December 4.

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.

Totaro, Paolo, and Domenico Ninno. 2014. "The concept of algorithm as an interpretative key of modern rationality." *Theory, Culture & Society:* 29–49.

Uprichard, Emma, Roger Burrows, and David Byrne. 2008. "SPSS as an 'inscription device': from causality to description?" *Sociological Review* 56 (4): 606–622.

Valiant, Leslie G. 1984. "A theory of the learnable." *Communications of the ACM* 27 (11): 1134–1142.

Van Dijck, José. 2012. "Facebook and the engineering of connectivity: A multi-layered approach to social media platforms." *Convergence: The International Journal of Research into New Media Technologies:* 1354856512457548.

Vance, Ashlee. 2009. "Data Analysts Captivated by R's Power." *The New York Times: Technology / Business Computing* (January 7).

———. 2011. "This Tech Bubble Is Different." *BusinessWeek: magazine* (April 14).

Vapnik, Vladimir. 1999. *The Nature of Statistical Learning Theory.* 2nd ed. 2000. Springer, December 1.

Venables, William N., and Brian D. Ripley. 2002. *Modern applied statistics with S.* Springer.

Venables, William, and B. D. Ripley. 2000. *S Programming.* Springer, April 20.

Virno, Paolo. 2004. *A grammar of the multitude : for an analysis of contemporary forms of life.* Semiotext(e) foreign agents series. Los Angeles: Semiotext(e).

What we're learning from online education | Video on TED.com. 2012. In collaboration with Daphne Koller. August.

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.

Wilf, Eitan. 2013. "Toward an Anthropology of Computer-Mediated, Algorithmic Forms of Sociality." *Current Anthropology* 54, no. 6 (December 1): 716–739. JSTOR: 10.1086/673321.

Wilson, Elizabeth A. 2010. *Affect and artificial intelligence.* University of Washington Press.

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