ADRIAN MACKENZIE

MACHINE LEARNERS: ARCHAEOLOGY OF A DATA PRACTICE

Preface

Although book is not an ethnography, 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. It's a case of 'dig where you stand;' or 'auto-archaeology.'

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

1. Typesetting the name of any code or devices that do machine learning and datasets on which machine learners operate in a `monospace` or terminal font: `machine learner` or `iris`;
2. Presenting formulae, functions, and equations using the bristling indexicality of mathematical typography: $\hat{\beta}$

I emulate the apparatus of science and engineering publication as an experiment in *in-situ* hybridization. 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 that of the participants in the research site they study. The object of study in this book, however, 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). It is possible for a social researcher to also adopt some of these practices.

I've been writing code for years (Mackenzie 2006). Writing code was nearly always something distant from writing about code since coding was about software projects and writing was about thinking and knowledge. I was slow to realise they are much entangled. Recent developments in ways of analysing and publishing scientific data bring coding and writing closer together. Implementing code can be done almost in the same space, in the same screen or pane, as writing about code. The mingling of coding and writing about code brings about sometimes generative, sometimes frustrating, encounters with various scientific knowledge (mathematics, statistics, computer science), with infrastructures and devices on many scales (ranging across networks, text editors, 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. In the interests of maintaining a strong feedback signal running through the many propositions, formulations, diagrams, equations, citations and images in this book, 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). As in the many machine learning textbooks, recipe books, cookbooks, how-tos, tutorials and manuals I have read, code, graphics and prose have been tidied here. Many exploratory forays are lost and almost forgotten.

The several years I have spent doing and writing about data practice has felt substantially different to any other project by virtue of the hybridization 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.

The capacity to mingle text, code and images depends on an ensemble of open source, often command-line 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, I have relied 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 writing (including mathematical formula) and images to be included, and importantly, executed together to produce a document.[1]

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 critical 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,

1. 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.

expression or performative utterance (Cox 2012; Coleman 2012), or more generally praxiography as a reality-making descriptive practice (Mol 2003) are right, 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.

Acknowledgments

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 with her almost geeky interest in genomics, Paul Oldham with his enthusiasm for ‘all the data,’ Maureen McNeil with her critical acuity, and Brian Wynne with his connective thought were participants in many discussions concerning the transformation of life sciences around which my interest in machine learning first ~~crystallised~~.

The Technology in Practice Group at ITU Copenhagen hosted some of the research work during 2014. Brit Ross Winthereik in particular made it possible for to develop some of the key ideas. I was lucky ~~too~~ to be part of an excellent research team on ‘Socialising ~~Big Data’(2013-2015)~~ that included Penny Harvey, Celia Lury, Ruth McNally and Evelyn Ruppert. We had excellent discussions.

My colleagues in the science studies at Lancaster, especially Maggie Mort, Lucy Suchman, and Claire Waterton are always a delight to work with. I have also been 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

provided inspiration in different ways. Sjoerd Bollebakker ~~very~~ kindly updated many of the scientific literature searches towards the end of the book's writing.

Various academic staff in the Department of Applied Mathematics and Statistics at Lancaster University shepherded me through ~~post-graduate~~ statistics 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.' Since 2015, I've also come to know some machine learners much better through the Data Science Institute, Lancaster University.

My inestimable friends in the *Computational Cultures* editorial group have listened to and irrationally encouraged various threads of work runnning the book. I warmly acknowledge the ~~existence~~ of Matthew Fuller, Andrew Goffey, Graham Harwood and Olga Goriunova. Nina Wakeford ~~at once point~~ encouraged me to be naïve.

Far away in Australia, Anna Munster has ~~stayed in touch~~. As always, Celia Roberts helped me sort out what I really want to do.

Contents

Acknowledgments 7

List of Figures 11

List of Tables 13

List of Code 14

1 *Introduction: Into the Data* 17

2 *Diagramming machines* 43

3 *Vectorisation and its consequences* 51

4 *Machines finding functions* 79

5 *N = ∀X Probabilisation and the Taming of Machines* 111

6 *Patterns and difference* 139

7 *Regularizing and materializing objects* 169

8 *Propagating subject positions* 201

9 *Conclusion: Out of the Data* 239

Glossary 255

Bibliography 259

Index 295

List of Figures

1.1 Google Trends search volume for machine learning 19

1.2 Cat as histogram of gradients 20

1.3 Machine learners in scientific literature 38

3.1 Scatter plot matrix of `prostate` data 63

3.2 Vector space comprises transformations 70

4.1 scikit-learn map of machine learners 80

4.2 Logistic or sigmoid function 92

4.3 South African Heart disease regularization plot 97

4.4 South African Heart disease decision plane 97

4.5 Gradient ascent for logistic regression 103

4.6 Stochastic gradient descent path 103

6.1 Recursive partitioning of the feature space 142

6.2 `AID classifier` 143

6.3 Decision tree on `iris` dataset 150

6.4 Support vector machine on `iris` dataset 155

6.5 MNIST Postal Digits 158

6.6 Margins in a support vector machine 159

7.1 A human genome diagrammed using ~~the~~ Circos 178

7.2 Hierarchical clustering of the SBRCT gene expression data 184

7.3 Shrinkage paths 191

7.4 Formulation of k-nn model 194

8.1 Techniques and concepts most frequently mentioned 207

8.2 The back-propagation algorithm 210

8.3 Neural network topology for 3-hidden unit ‘titanic‘ data 222

8.4 Kaggle data science competitions 230

List of Tables

1.1 A small sample of titles of scientific articles that use machine learning in relation to "difference" 7

2.1 The truth table for the Boolean function NOT-AND truth 47

3.1 Datasets in *Elements of Statistical Learning* 54

3.2 First rows of the 'prostate' dataset 63

3.3 Fitting a linear model to the extttprostate dataset 73

4.1 Sample of highly cited machine learning publications referring to "function" in title or keyword 108

4.2 Sample of highly cited scientific publications referring to "logistic regression" in title or keyword 109

5.1 Some structuring differences in machine learning 117

5.2 Most cited Naive Bayes publications 1945-2015 127

6.1 References to Morgan and Sonquist's Automatic Interaction Detector 143

6.2 Most cited papers on support vector machines 155

7.1 The top 20 disciplines of the top 5000 cited research publications in machine learning, 1990-2015 176

7.2 First 5 rows of Fisher's 'iris' dataset 181

7.3 Small round blue-cell tumor data sample (Khan, 2001) 183

8.1 The highest prize money machine learning competitions on Kaggle 233

List of Code

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~~ yea  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. The advances, particularly over the last five years, have been startling (Breiman 2001b, 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 said to "learn from experience" has become operational in the last three decades. Known by various ~~names — machine~~ learning, pattern recognition, knowledge discovery, data ~~mining — the~~ field and its devices, which all take shape as computer programs or code, seem to have quickly spread across scientific disciplines, business and commercial settings, industry, engineering, media, entertainment, and government. Heavily

dependent on computation, 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, ornithology, finance, surveillance (see the U.S. Government's `SkyNet` for one example of a machine learning surveillance system (Agency 2012)) or robots(see a Google robotic arm farm learning to sort drawers of office equipment such as staplers, pens, erasers and paper clips (Levine et al. 2016).

Sometimes machine learning devices are understood as *scientific models*, and sometimes they are understood as *operational algorithms*. 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 many biomedical papers). They anchor the field of 'data science' (Schutt and O'Neil 2013), as institutionalised in several hundred data science institutes scattered worldwide. Not so recently, they also became mundane mechanisms deeply embedded in other systems or gadgets (as in the decision tree models used in some computer game consoles to recognise gestures, the neural networks used to recognise voice commands by search engine services such as `Google Search` and `Apple Siri` (McMillan 2013) or Google's `TensorFlow` software packages that puts deep convolutional neural nets on Android devices (Google 2015)). In platform 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 news feeds on Facebook.

In all of these settings, applications and fields, machine learning is said to transform the nature of knowledge. Might it transform

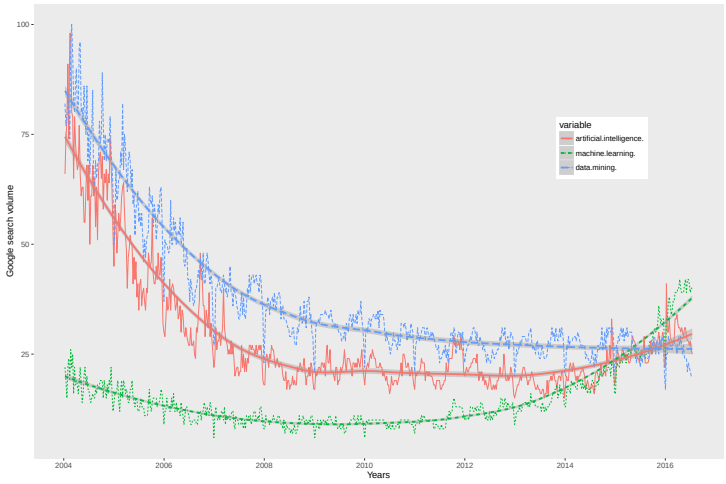

Figure 1.1: Google Trends search volume for 'machine learning' and related query terms in English, globally 2004-2016

the practice of critical thought? This book is an experiment in such practice.

Three accumulations: settings, data and devices

Three different accumulations cross-stratify in machine learning: settings, data and devices. The volume and geography of searches on Google Search provides some evidence of the diverse settings or sites doing machine learning. If we search for terms such as `artificial intelligence`, `machine learning` and `data mining` on the Google Trends service, the results for the last decade or so suggest shifting interest in these topics.

In Figure 1.1, two general search terms that had a 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` loses volume until around 2008, and then gradually rises again so that by mid-2016 it exceeds the long-standing interests in data mining and artificial intelligence. Whatever the difficulties in understanding GoogleTrends results, these

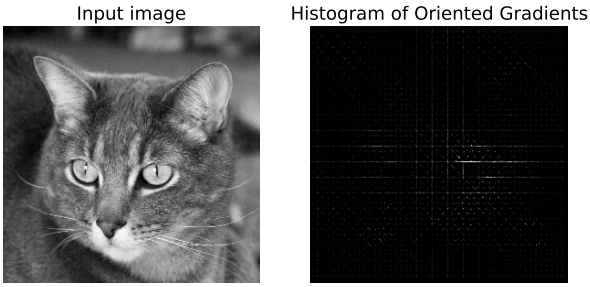

Figure 1.2: Close up of cat. The image on the left is already signal-processed as a JPEG format file. The image on the right is further processed using histogram of oriented gradients (HOG) edge detection. `kittydar` models HOG features. Cat photo courtesy photos-public-domain.com

curves suggest an accumulation of sites and settings turning towards machine learning.[1] What does it mean that machine learning surfaces in so many different places, from fMRIs to Facebook's `AI-Flow` (**Facebook_2016**), from fisheries management to Al Queda courier network monitoring by `SkyNet`?[2]

A second accumulation concerns the plenitude of things in the world as data. If we wanted to describe the general horizon of machine learning as a data practice in its specificity, we might turn to cats. Cat images accumulate on websites and social media platforms as de-centred, highly repetitive data forms. Like the billions of search engine queries, email messages, tweets, or much contemporary scientific data (e.g. the DNA sequence data discussed in chapter 7), images accumulate in archives of communication. 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' (H. Arthur 2012). This playful piece of code demonstrates how machine learning can be amidst mundane accumulation. Heather Arthur, who developed `kittydar` writes:

> Kittydar first chops the image up into many "windows" to test for

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 events such as significant press releases, public debates, media attention and film releases. 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.

2. The diagram shown in Figure 1.1 actually draws 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 irregular variations with a curve that unequivocally runs through time with greater regularity. The smoothed lines shade the diagram with a predictive pattern. The lineaments of machine learning already appear in such lines. How have things been arranged so that smooth lines run through an accumulated archive of search data?

> 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 … 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 … which gives a number from 0 to 1 on how likely the histogram data represents a cat. The neural network … 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 (H. Arthur 2012).

This toy device finds cat heads in digital photographs, but also exemplifies many key traits of machine learning. Large accumulations of things become vectors a dataset. A dataset is used to train a typical machine learning device, a neural net, and the neural net *classifies* subsequent images probabilistically. The code for all this is 'in the repo.' Based on how it locates cats, we can begin to imagine similar pattern recognition techniques in use in self-driving cars (Thrun et al. 2006), border control facial recognition systems, military robots or wherever something seen implies something to do.

Faced with the immense accumulation of cat images on the internet, `kittydar` can do very little. It only detects the presence of cats that face forward. And it sometimes classifies people as cats. 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 – a spatial order of great significance in machine learning – running from light and dark, and then compares these measurements to the gradients of known cat images (the so-called 'training data'). 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 8, 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 6).

A final accumulation comprises machine learning techniques and devices. Machine learning range from the mundane to the esoteric, from code miniatures such as `kittydar` to the infrastructural sublime of computational clouds and clusters twirling in internet data-streams. Like the images that `kittydar` classifies, the names of machine learning techniques and devices proliferate and accumulate 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, 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. Intricate data practices — normalization, regularization, cross-validation, feature engineering, feature selection, optimisation — embroider 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 disparate 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. Machine learning itself is an accumulation rather than a radical transformation.

Who or what is a machine learner?

I am focusing on machine ~~learners — a~~ term that refers to both humans and machines or human-machine relations throughout this ~~book — situated~~ amid~~st~~ these three accumulations of settings, data and devices. ~~While~~ it is not always possible to disentangle machine learners from the databases, infrastructures, platforms or interfaces they work through, I will argue that data practices associated with machine learning delimit a *positivity* of knowing. The term 'positivity' comes from Michel Foucault's *The Archaeology of Knowledge* (Foucault 1972)~~.~~ and refers to specific forms of accumulation of statements grouped in a discursive practice. Analyzed arch~~a~~eologically, a positivity can be investigated and inhabited to some degree by critical thought.

Foucault attributes a lift-off effect to positivity:

> The moment at which a discursive practice achieves individuality and autonomy, the moment therefore at which a single system for the formation of statements is put into operation, or the moment at which this system is transformed, might be called the threshold of positivity~~.~~ (186)

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: what happen as machine learners shift from localized 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 settings the pronounced uptick shown in Figure 1.1 attest to a re-definition of knowledge, decision and control, a new operational formation in which 'a system is transformed'?

Algorithmic control to the machine learners?

Written in code, machine learners operate as programs or computational processes to produce statements that may take the form of numbers, graphs, propositions (see for instance the propositions produced by a recurrent neural net on the text of this book in the concluding chapter 9). Machine learning can also be viewed as a change in how programs or the code that controls computer operations are developed and operate (see chapter 2 for more detailed discussion of this). The term 'learning' in machine learning points to this change and many machine learners emphasize it. Pedro Domingos, for instance, a computer scientist at the University of Washington, writes:

> Learning algorithms — also known as learners — are algorithms that make other algorithms. With machine learning, computers write their own programs, so we don't have to. (Domingos 2015, 6)

Viewed from the perspective of control, and how control is practiced, digital computer programs stem from and epitomise the 'control revolution' (Beniger 1986) that arguably has, since the late nineteenth century, programmatically re-configured production, distribution, consumption, and bureaucracy by tabulating, calculating and increasingly communicating events and operations. With the growth of digital communication networks in the form of the internet, the

late 20th century entered a new crisis of control, no longer centred on logistic acceleration but on communication and knowledge. Almost all accounts of the operational power of machine learning emphasise its power to inflect the control of processes of communication — border flows, credit fraud, spam email, financial market prices, cancer diagnosis, targeted online adverts — processes whose unruly or transient multiplicity otherwise evades or overwhelm us — with knowledge (classifications and predictions in particular) derived from algorithms that make other algorithms. On this view, `kittydar` can isolate cats amidst the excessive accumulation of images on the internet because neural net learning algorithms (back-propagation, gradient descent) have written a program — 'a pre-trained' neural net — during its training phase.

If a newly programmatic field of knowledge-control takes shape around machine learning, how would we distinguish it from computation more generally? Recent critical research on algorithms offers one lead. 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 calculations automate decisions. 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 (as we will see in exploring algebraic operations in chapter 3, in finding and optimising certain mathematical functions in chapter 4 or in characterising and shaping probability distributions in chapter 5). It works to mediate future-oriented decisions (although all too often, very near-future decisions such as ad-click prediction).

I am less certain about treating machine learning as automation. Learning from data, as we will see, does often sidestep and substitute for existing ways of acting, and practices of control, and it thereby re-configures human-machine differences. Yet the notion of automation does not capture well how this comes about. The programs that machine learner 'write' are formulated as probabilistic models, as learned rules or association, and they generate predictive and classificatory statements ('this is a cat'). They render calculable some things which hitherto appeared intractable to calculation (for instance, the argument of a legal case). Such calculation, with all the investment it attracts (in the form of professional lives, in the form of infrastructures, in reorganisation of institutions, corporations and governments, etc.) does rule out some and reinforce other futures.[3] If this transformed calculability is automation, we need to understand the specific contemporary reality of automation as it takes shape in machine learning. We cannot conduct critical enquiry into the calculation that will automate future decisions without opening the very notions of calculation and automation into question.

Does the concept of algorithm help us identify the moments that come together in machine learning without resorting to a-historical concepts of automation or calculation? In various scholarly and political debates around changes business, media, education, health, government or science, quasi-omnipotent agency has been imputed

3. 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. 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).

to algorithms (Barocas, Hood, and Ziewitz 2013; Beer and Burrows 2013; Cheney-Lippold 2011; Fuller and Goffey 2012; A. R. Galloway 2004); Gillespie (2014); Neyland (2015); Pasquinelli (2014); Smith (2013); Totaro and Ninno (2014); Wilf (2013)] or sometimes just 'the algorithm'. This growing body of work understands the power of algorithms in the social science and humanities literature in different ways, sometimes in terms of rules, sometimes as functions or mathematical abstractions, and increasingly as a located practice. There is general agreement that algorithms are powerful, or at least, can bear down heavily on people's lives and conduct, re-configuring, for instance, culture as algorithmic (Hallinan and Striphas 2014).

Some of the critical literature on algorithms identifies abstractions 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 for critical practice, and for experiments in such practice. Since in mathematics let alone critical thought, abstraction can be understood in many different ways, any direct identification of algorithms with abstraction will, however, be difficult to practice. Which algorithm, what kind of abstraction and which 'mathematical way' should we focus on? Like automation and calculation, abstraction and mathematics have mutable historicities. We cannot 'move at the same level' without taking that into account.

Furthermore, given the accumulations of settings, data and devices, there might not be any single level of abstraction to move at, only 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.

The archaeology of operations

Given mathematics and algorithms do loom large in machine learning, how do we address their the workings without pre-emptively ascribing potency to mathematics, or to algorithms? In the chapters that follow, I do explore specific learning algorithms (gradient descent in chapter 4 or recursive partitioning 6) and mathematical techniques (the sigmoid function in chapter 4 or inner products in chapter 3) in greater empirical and conceptual depth. Following much scholarship in science and technology studies, I maintain that attention to specificity of practices is an elementary prerequisite to understanding human-machine relations, and their transformations. The archaeology of operations that I will develop combines an interest in machine learning as a form of knowledge production and a strategy of power. Like Foucault, I see no exteriority between techniques of knowledge and strategies of power ('between techniques of knowledge and strategies of power, there is no exteriority, even if they have specific roles and are linked together on the basis of their difference' (Foucault 1998, 98)).

If we understand machine learning as a data practice that re-configures local centres of power-knowledge through a re-drawing of human-machine relations, then the specific roles and differences associated with machine learners in the production of knowledge should be a focus of attention. Differences are a key concern here

since many machine learners classify things. They are often simply called 'classifiers.' Some of the practice of difference works in terms of categories. `Kittydar` classifies images as `cat` with some probability, but categorisation and classification in machine learning occurs much more widely.[4] We might understood the importance of categories sociologically. For instance, 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).

Orderings of categorical differences undergo a great deal of intensification via machine learning. Categories are often simply an existing set of classifications derived from institutionalised or accepted knowledges (for instance, the categories of customers according to gender or age). Machine learners also generate new categorical workings or mechanisms of differentiation. As we will see (for instance in chapter 7 in relation to scientific data from genomes), machine learners invent or find new sets of categories for a particular purpose (such as cancer diagnosis or prognosis). These differentiations may or may not bring social good. The person who finds themselves paying a higher price for an air ticket by virtue of some unknown combination of factors including age, credit score, home address, previous travel, or educational qualifications experiences something of the classificatory power.

4. 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' as an operational field of knowledge production.

Asymmetries in common knowledge

What can critical thought, the kind of enquiry that seeks to identify the conditions that concretely constitute what anyone can say or think or do, learn from machine learning? 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 (propagating and proliferating in many domains and settings) and stratification (the socially, epistemically, economically and sometimes politically or ontologically loaded re-iterative enactment of categories). I am not sure that understanding how a support vector machine or a random forest orders differences would change how ~~would we~~ relate to what we see, feel, sense, hear or think in the face of a contemporary platform such as Amazon's ~~that~~ uses Association Rule Mining -, an app, a passport control system 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).

Machine learners ~~themselves~~ sometimes complain of the monolithic and homogeneous success of machine learning. Some expert practitioners complain of a uniformity in its applications. 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 best of my generation are thinking about how to make people click ads' (Vance 2011)~~–~~. Leaving aside debates about the ranking of 'best minds' (a highly competitive and exhaustively tested set of subject positions; see chapter 8), Hammerbacher was lamenting 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.

Hammerbacher's unhappiness with ad-click prediction resonates with critical responses to the use of machine learning in the digital humanities. Some versions of the digital humanities make extensive use of machine learning. To cite one example, in *Macroanalysis: Digital Methods and Literary History*, Matthew Jockers describes how he relates to one currently popular machine learning or statistical modelling technique, the topic model (itself the topic of discussion in Chapter 5; 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 Latent Dirichlet Allocation or 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 accumulations of documents (scientific publications, news, literature, online communications, etc.) in terms of underlying themes or latent 'topics.' As we will, large document collections have long attracted the interest of machine learners (see chapter 5). On

the other hand, Jockers signals the practical 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 some parts of the humanities and critical social research exhorts closer attention to algorithms and mathematical abstractions, other parts elides its complexity in the name of 'the beauty of the results.'

Critical thought has not always endorsed the use of machine learning in digital humanities. Alex Galloway makes two observations about the circulation of these methods in humanities scholarship. The first points to its marginal status in increasingly machine-learned media cultures:

> 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 (A. Galloway 2014, 110)

Galloway highlights the asymmetry between humanities scholars and media enterprises or credit score agencies (Equifax). 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.

Galloway also observes that even if 'cultural workers' do manage

to learn to machine learn, and become adept at re-purposing the techniques in the interests of analyzing culture rather than selling things or generating credit scores, they might actually reinforce power asymmetries and exacerbate the ethical and political challenges posed by machine learning:

> 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 concerning the nature of any knowledge produced by machine learning. The 'techniques' of machine learning may or may not be positivist, and indeed, given the claims that machine learning transforms the production of knowledge, positivism may not be any more stable than other conceptual abstractions. Hence, it might not be so strongly at odds with critical thought, even if remains complicit 'aligned' with capitalist work. Intellectual work of the kind associated with machine learning is definitely at the centre of many governmental, media, business and scientific fields of operation and increasingly they anchor the operations of these fields. Yet neither observation asymmetries in scale, alignment with a 'positivist' capitalist knowledge economy exhaust the potentials of machine learning, particularly if, as many people claim, it transforms the nature of knowledge production and hence 'brainwork.'

What cannot be automated?

Jaron Lanier's ~~question — how~~ will we conceive at a given time what cannot be automated?~~ — suggests~~ an alternative angle of approach. Like Galloway, I'm wary of certain deployments of machine learning, particularly the platform-based media empires and their efforts to capture sociality (Gillespie 2010; Van Dijck 2012). Machine learners do seem to be 'laying claim to the apparatus of knowledge production.' Yet even amid~~st~~ the jarring ephemera of targeted online advertising or the more elevated analytics of literary history, the transformations in knowledge and knowing do not automatically appropriate intellectual work to capitalist production. Empirical work to describe differences, negotiations, modifications and contestation of knowledge would be needed to show the unevenness, variability and deep contingency of that appropriation. As I have already suggested, machine learning practice is not simply automating existing economic relations or even data practices. ~~While~~ Hammerbacher and Galloway are understandably pessimistic about 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.

Importantly, the familiar concerns of critical social thought to analyse differences, power, materiality, subject positions, agency, ~~etc.~~ somewhat overlap with the claims that machine learning produces knowledge of differences, of nature, cultural processes, communication and conduct. Unlike other objects of critical thought, machine learners (understood always as human-machine relations) are ~~themselves~~ closely interested in producing knowledge, albeit scientific, governmental or operational. This coincidence of knowledge projects suggests

the possibility of some different articulation, of modification of the practice of critical thought in its empirical and theoretical registers. The altered human-machine relations we see as machine learners might shift and be re-drawn through experiments in empiricism and theory.

Where in the algorithms, calculations, abstractions and regularizing practices of machine learning would differences be re-drawn? 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 supports a situationally aware calculative ~~knowledge-practice~~.

For instance, the campaign to re-elect Barack Obama as U.S. ~~P~~president in ~~2011-12~~ relied heavily on micro-targeting of voters in the lead up to the election polls (Issenberg 2012; Mackenzie et al. 2016). In response to the data analytics-driven election campaign run by the ~~US~~ Democrats, data journalists at the non-profit news organisation *ProPublica* reverse engineered the machine learning models that the Obama re-election team used to target individual votes with campaign messages (Larsen 2012). They built their own machine learning model ~~-~~the 'Message ~~Machine' - using~~ emails sent in by voters to explore the workings of the Obama campaign team's micro-targeting 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-targeting

processes.

This reverse engineering work focused on the constitution of subject positions (the position of the 'voter') 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.

Both cases explore the constitutive contemporary conditions of doing, saying, and thinking of subjects, not only to recognise how subject positions are assigned or made, but to grasp the possibility of change. While these cases may be exceptional achievements, and indeed highlight the dead weight of ad-tech application of machine learning, knowledge production more generally is not easily reducible to contemporary forms of capitalism labour.

Different fields in machine learning?

The proliferation of scientific machine learners suggests that the generalization of machine learning cannot be reduced to personalized advertising or other highly extractive uses. Table 1.1 presents a small sample of scientific literature at the intersection of 'difference' and machine learning. This sample, while no doubt dwarfed by the flood of computer science publications on recommendation systems, targeted advertising or handwriting recognition, is typical of the positivity or specific forms of accumulation associated with

Title	Year
iDiff: Informative summarization of differences in multidimensional aggregates	2001
Why voxel-based morphometric analysis should be used with great caution when characterizing group differences	2004
Empirical bounds on error differences when using Naive Bayes	2005
Multivariate adaptive regression splines: a powerful method for detecting disease-risk relationship differences among subgroups	2006
Analysis of categorical response data: Use logistic regression rather than endpoint-difference scores or discriminant analysis	2009
Beta-MPT: Multinomial processing tree models for addressing individual differences	2010
Clustering and variance maps for cryo-electron tomography using wedge-masked differences	2011
New Theoretical Results on Channelized Hotelling Observer Performance Estimation With Known Difference of Class Means	2013
Differences in gut microbiota composition between obese and lean children: a cross-sectional study	2013
Differences in cognitive aging: typology based on a community structure detection approach	2015

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

machine learners in science. (I return to this topic in Chapter 7 in discussing how the leveraging of scientific data via predictive models and classifiers deeply affects the fabric and composition of objects of scientific knowledge.) 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.

Similarly, if we attend to the flow of machine learning practices, devices and techniques in scientific fields, a diversification rather than a simple scaling-up to industrial-strength infrastructures begins to appear. Figure 1.3 derives from counts of scientific publications that mention particular machine learners such as `decision tree` or `Naive Bayes` in their title, their abstract or keywords. The curves, which are probability density plots, suggest a time-varying distribution of statements and operations for different techniques. This crude plot outlines the duration and the ebbs and flows of work on specific techniques, platforms, knowledges and power relations. Like the Google Trends searches for `machine learning`, the lines shown in Figure 1.3 have been normalised in order to adjust for an overall

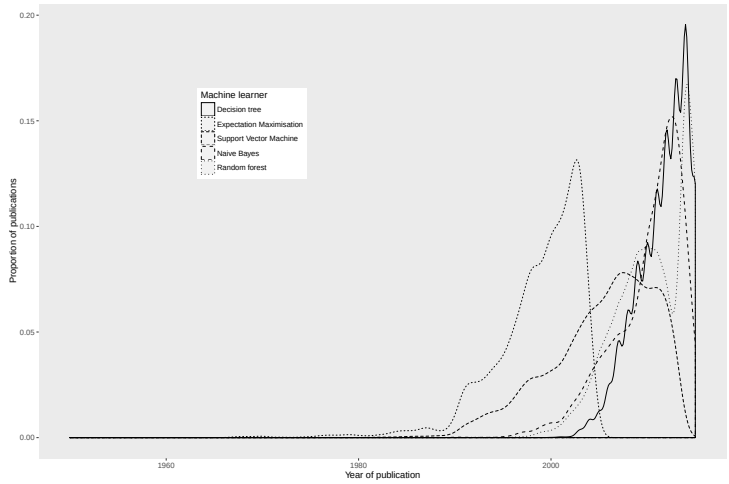

Figure 1.3: Machine learners in scientific literature. The lines in the graph suggest something of the changing fortunes of machine learners over time. The publication data comes from Thomson Reuter's *Web of Science*. Separate searches were run for each machine learner. In these plots, as in the ~~GoogleTrends~~ 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.

increase in the volume of scientific publications over the last five decades. Unlike the Google Trends search patterns, the scientific literature displays polymorphous temporalities in which different techniques and operations diverge widely from each other over the last half century. Crucially for my purposes, machine learning in the sciences constitutes an a-totality, a heterogeneous volume and ~~de-centred~~ production of statements.

The diagram in critical thought

As an experiment in the practice of critical thought amid~~st~~ the accumulations of data, settings and devices, this book attempts to diagram the data practices of machine learning ~~in~~ respect to knowledge production. Despite their many operational deployments, the coming together of algorithm, calculation and technique in machine learning 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 or algorithmic essence to them nor frames them as means of production of relative surplus value. Finding ways of accommodating their diversity, loose couplings and mutability would mean grasping their operational power and their potential to create new forms of difference.[5]

Diagrams — a form of drawing that smooths away many frictions and variations — practically abstract. (Gilles Deleuze, in his account of Michel Foucault's philosophy, presents diagrams as centres of power-knowledge: 'What is a diagram? It is a display of the relations between forces which constitute power in the above conditions … the diagram or abstract machine is the map of relations between forces, a map of destiny, or intensity' (Deleuze 1988b, 36)). Diagrams retain a connection to forms of doing, such as 'learning from experience,' that idea-centric accounts of abstraction sometimes struggle with. Perceptually and operationally, they span and indeed criss-cross between human and machine machine learners. As we will see, they form an axis of human-machine relations in machine learning, and a significant site of invention and emergence. They accommodate compositional substitutions, variations and superimpositions, as well as a play or movement amongst their often heterogeneous elements. Occasionally, perhaps rarely (Foucault as we will see characterises statements by their rarity amidst accumulation), by virtue of its composition, diagrams bring something new into the world.

Both the commonality and specificity (indeed the specific commonality) of machine learners would be hard to grasp without being able to trace their diagrammatic composition. Similarly, in order to understand the operational formation associated with machine learning, the connections between data structures, infrastructures,

5. 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.

processors, databases and lives need to be mapped. One path to follow in doing this - certainly not the one for everyone – is to inhabit and recognise oneself as a machine learner by occupying associated subject positions (the programmer or software developer, the statistics or computer science student, the modeller, the researcher, the data scientist, etc.). In moving between some of these subject positions, the densely operational indexes of mathematical formalisms begin to unravel.

Diagrams can be drawn in multiple ways, using various materials and inscriptive practices. Perhaps naively interpellated by the claim of machine learning to know differently, I use code and software implementations, graphical plots and mathematical expressions absorbed or copied from textbooks, blogs, online videos and the heavy accumulation of scientific publications from many disciplines (for instance, as seen in figure 1.3), together with the theoretical resources of a media-focused archaeology of knowledge and a science studies-informed ethnographic sensibility towards always situated and configured infrastructural and calculative practices.

From my own learning to machine learn, I draw six major machine learning operations diagrammatically: vectorisation, optimisation, probabilisation, pattern recognition, regularization and propagation. These generic operations intersect in 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, and professional and popular pedagogies. With varying degrees of formalization and consistency, these operations might also occasion or provoke some creative, resistive or re-distributive moves for critical thought in relation to differences, materiality, experience, agency or power. They

are the topics of chapters 3-??. A summary of the operations can also be found near the beginning of the concluding chapter.

My somewhat risky a-critical immersion in technical practice seeks to support an alternative account of machine learning, an account in which some feeling of agency movement can take root. Mundane technical practices, sometimes at a quite low level (for instance, vectorisation) and other times at a high level of formalization (for instance, in discussing mathematical functions) are elements to be drawn – sometimes literally, sometimes operationally – on a diagram. The archaeology of the operational formation of machine learning does not unearth 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 accumulations and peripheral variations.

2

Diagramming machines

> 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)

> The tools or material machines have to be chosen first of all by a diagram (Deleuze 1988b, 39)

The 'learning' in machine learning embodies a change in programming practice or indeed the programmability of machines. Our sense of the potentials of machine learning can be understood, according to Pedro Domingos, in terms of a contrast between programming as 'a lot of [building] work' and the 'farming' done by machine learners to 'grow programs'. In characterising machine learning, the tensions between the programming 'we' (programmers, computer scientists?) do and the programming that learners do ('growing') are worth pursuing. While Domingos suggests that machine learners 'get more from less,' I will propose that an immense constellation of documents, software, publications, blog pages, books, spreadsheets, databases, data centre architectures, whiteboard and blackboard drawings, and

an inordinate amount of talk and visual media orbit around machine learning. There has been lively growth in machine learning, but this liveliness and the sometimes life-like growth of machine learners are a regional expression of a distributed formation. 'Wherever there is a region of nature,' the philosopher Alfred North Whitehead writes, 'which is itself the primary field of the expressions issuing from each of its parts, that region is alive' (Whitehead 1956, 31).

In this chapter I attend to the problem of identifying and describing the distributed practices that give rise to a sense of machine learners framed as growth, or liveliness. I will argue these practices can only be traced partially through code written in generic or specialized programming languages such as `Python` and `R`, in libraries of machine learning code such as `R`'s `caret` or `Python`'s `scikit-learn` or `TensorFlow` to do machine learning. Code obscures and reveals multiple transformations at work in the operational formation. Science studies scholars such as Anne-Marie Mol have urged the need to keep practice together with theories of what exists. 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 (Mol 2003, 6).

Mol advocates thinking reality as multiplicity. Her insistence on the nexus of practice or doing and the plural existence of things suggests a way of handling the code that machine learners produce. Code should be approached multiplicity. In the case of machine learners, this means following through pedagogical expositions of machine learning focused on both mathematical derivations and the accumulation of scientific or technical research publications, ranging from textbooks to research articles, that vary, explore, experiment

and implement machine learners in code. The effect of liveliness or growth issues from many parts. 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.

While machine learning utterly depends on code, I will suggest therefore that no matter how expressive or well-documented it may be, code alone cannot fully diagram how machine learners make programs or how they combine knowledge with data. Domingos writes that 'learning algorithms … are algorithms that make other algorithms. With machine learning, computers write their own programs, so we don't have to' (Domingos 2015, 6). Yet the writing performed by machine learners cannot be read textually or procedurally as other programs might be read for instance in work known as critical code studies. The difference between the reading of an Atari computer game console in Nick Montfort and Ian Bogost's *Racing the Beam* (Montfort and Bogost 2009) and the machine learning of Atari's games undertaken by DeepMind in London during recent years (Mnih et al. 2013, 2015) is hard to read from program code. The learning or making by learning is far from homogeneous, stable or automatic in practice. Materially, code is only one element in the diagram of machine learning. It displays, with greater or lesser degrees of visibility relations between a variety of forces (infrastructures, scientific knowledges, mathematical formalisations, etc.). It itself is aligned by and exposes other institutional, infrastructural, epistemic and economic positions.

Coding practices and the pedagogical expositions of machine learning 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. The fact that data scientists, software developers and other machine learners across scientific and commercial settings use programming languages such as `Python` and `R` more than specialized commercial statistical and data software packages such as `Matlab`, `SAS` or `SPSS` (Muenchen 2014) is perhaps symptomatic of shifts in computational culture. Coding cultures are crucial to the recent growth of machine learning. Although scientific computing languages such as `FORTRAN` ~~–~~'Formula ~~Translator' have~~ long underpinned scientific research and engineering applications in various fields (Campbell-Kelly 2003, 34–35), the development in recent decades of data-analytic and statistical programming languages and coding frameworks has crystallized a repertoire of standard operations, patterns and functions for reshaping data and constructing models that classify and predict events and associations ~~between~~ things, people, processes, ~~etc.~~ This development continues apace, especially in research and engineering driven by social media and internet platforms such as Facebook and Google. ~~While~~ Domingos speaks of 'growing' programs, the accumulating sediment of certain well-established data practices are the soil in which programs take root. The different elements of coding practice are precisely the faceted levels of abstraction ~~which~~ we need to access and traverse ~~in order~~ to know and come to grips empirically with contemporary compositions of power and knowledge in machine learning.

'We don't have to write programs'?

In machine learning, coding changes from what we might call symbolic logical diagrams to statistical algorithmic diagrams. ~~While~~ many

machine learning techniques have long statistical lineages (running 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 often embody a certain dissatisfaction with the classical computer science understanding of programs as manipulation of symbols, even as they rely on such symbolic operations to function. Symbolic manipulation, 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 machine learning, the privileged symbolic-cognitive forms of logic are subject to a 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 2.1. The truth table summarises a logical function that combines three input variables `X1`, `X2`, and `X3` and produces the output variable `Y`. Because in Boolean calculus, variables or predicates can only take the values `true` or `false`, they can be coded in as `1` and `0`.

Table 2.1: The truth table for the Boolean function NOT-AND truth

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

Now in Foucaultean terms, the truth table and its component propositions constitutes a statement. This statement has triple relevance for archaeology of machine learning. The spatial arrangement of the table is fundamental (and this is the topic of chapter 3). 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. The numbers 1 and 0 are the binary digits as well as the 'truth' values 'True' and 'False' in classical logic. These numbers are readable as symbolic logical propositions governed by the rule $Y = \neg X_1 \qquad X_2$

3

Vectorisation and its consequences

> The table has the function of treating multiplicity itself, distributing it and deriving from it as many effects as possible (Foucault 1997, 149)

> We call *any* set that satisfies these properties (or axioms) a *vector space*, and the objects in the set are called *vectors*. (Larson 1996, 166)

> All things are vectors (Whitehead 1960, 309)

The operational power of machine learning locates data practice in an expanding epistemic space. The space derives, I will suggest, from a specific operational diagram that maps data into a vector space. It *vectorizes* data according to axes, coordinates, and scales. Machine learning, in turn, inhabits a vectorised space, and its operations vectorise data.

Often data is represented as an homogeneous mass or a continuous flowing stream. My aim here, however, is to archaeologically examine some of the transformations that allow different shapes and densities of data, whether in the form of numbers, words or images, to become machine learnable. Data in its local complexes spaces out in

many different density shapes, depending on how the data has been generated or instanced.[1] Whatever the starting point — a measuring instrument, people clicking and typing on websites, a device like a camera, a random number generator, etc. — machine learners only ever encounter data in specific *vectorised* shapes (vectors, matrices, arrays, etc.), mapped to a geometrically coordinate volume. The mapping and forming, when mentioned at all, is sometimes referred to as 'data cleaning' but that term covers over important but largely taken for granted and constitutive transformations. The archaeology of data shapes presented in this chapter explores a range of transformations focused around the table or row-column grid.

1. I loosely borrow the term 'density' from statistics, where *probability density functions* are often used to describe the hardly ever uniform distribution of probabilities of different values of a variable. Sensing density as a form of variation matters greatly both in machine learning itself, where algorithms seek purchase on unevenly distributed data, and in any broader diagram of data. Probability densities are discussed in much more detail in Chapters 5. The collection *"Raw Data" is an Oxymoron* (Gitelman 2013) evinces some of these different densities and distributions of data.

The reshaping and re-flowing of data densities into vectors deeply affects machine learning. This forming and reforming of data is evidence of implicated relations. The philosopher A.N. Whitehead called 'strain':

> a feeling in which the forms exemplified in the datum concern geometrical, straight, and flat loci will be called a "strain." In a strain, qualitative elements, other than the geometrical forms, express themselves as qualities implicated in those forms (Whitehead 1960, 310).

In many machine learning models the exemplified forms are straight or flat loci (as we see in chapter 4). Yet different practices also elicit relations that strain the linear shaping of data and these divergent relations sometimes combine in generative and provocative ways.

Vector space and geometry

Statistical modelling, data-mining, pattern recognition, recommendation systems, network modelling and machine learning rely very much on the operation called 'fitting a model'. Fitting as a spatial prac-

tice has some elements that resemble the phenomenologist Edmund Husserl's account of the origin of geometry. Husserl writes:

> First to be singled out from the thing-shapes are surfaces — more or less "smooth," more or less perfect surfaces; edges, more or less rough or fairly "even"; in other words, more or less pure lines, angles, more or less perfect points; then again, among the lines, for examples, straight lines are especially preferred, and among surfaces, the even surfaces. … Thus the production of even surfaces and their perfection (polishing) always plays its role in praxis (Derrida 1989, 178)

Husserl here refers is attempting to describe something of the way in which forms such as planes, lines, circles, triangles, squares, and points became objects of geometrical practice. A similar polishing and smoothing of surfaces is certainly taking place today in the thing-shapes we call data. The basic machine learning work of 'fitting a model' (or many models) to data is often literally implemented, as we will see, by constraining data within a coordinate, discretized space, which I term the 

Critical thought from phenomenology to social theory has a long-standing nervousness about the power of geometry and its gradual movement away from shapes and things towards mathematical operations. The philosopher Hannah Arendt, for instance, observes:

> decisive is the entirely un-Platonic subjection of geometry to algebraic treatment, which discloses the modern idea of reducing terrestrial sense data and movements to mathematical symbols (Arendt 1998, 265)

The crux of the problem rests on the 'treatment' or operations that 'reduce terrestrial sense data and movements' to symbols. One challenge for contemporary thought is how to orient itself to such operations, particularly in their data-intensive forms, without assuming that the familiar story of scientific and technical reduction

of sense and movement to the lines and planes of modern geometry is simply reinforced in contemporary data practice. If an archaeology of data vectorization does anything, it needs to offer an alternative to that reduction.

They also, as we will see, reach down into the practices of programming, infrastructure and hardware production in ways that differ somewhat from increases in computational power or speed. Familiar narratives of Moore's Law increases in speed or efficiency of computation do not account for transformations of data in `R` and other computing environments (~~for instance,~~ the popular `Map-Reduce` architecture invented at Google Corporation to speed up its search engine services (Mackenzie 2011)). Vectorisation transforms data along more diagrammatic lines.[2]

2. Later chapters will discuss various ways in which the vectoral dimensionality of data or its rendering as *vector space* scales up and ~~scales~~ down in machine learning. In terms of multiplying matrices, dimensionality both constrains and enables many aspects of the prediction. Perhaps on the grounds of data dimensionality alone, we should attend to dimensionality practices in `R`. A fuller discussion of dimensionality of data is the topic of chapter 6. There I discuss how machine learning re-dimensions data in various ways, sometimes reducing dimensions and at other times, multiplying dimensions.

Mixing places

	Item	Title
1	SAheart	South African Hearth Disease Data
2	bone	Bone Mineral Density Data
3	countries	Country Dissimilarities
4	galaxy	Galaxy Data
5	marketing	Market Basket Analysis
6	mixture.example	Mixture Example
7	nci	NCI microarray data (chap 14)
8	orange10.test	Simulated Orange Data
9	orange10.train	Simulated Orange Data
10	orange4.test	Simulated Orange Data
11	orange4.train	Simulated Orange Data
12	ozone	Ozone Data
13	phoneme	Data From a Acoustic-Phonetic Continuous Speech Corpus
14	prostate	Prostate Cancer Data
15	spam	Email Spam Data
16	vowel.test	Vowel Recognition (Deterding data)
17	vowel.train	Vowel Recognition (Deterding data)
18	waveform.test	Simulated Waveform Data
19	waveform.train	Simulated Waveform Data
20	zip.test	Handwritten Digit Recognition Data
21	zip.train	Handwritten Digit Recognition Data

Table 3.1: Datasets in *Elements of Statistical Learning*

Data appears in *Elements of Statistical Learning* in multiple forms. Maps of New Zealand fishing patterns lie next to plots of factors in South African heart disease. The 21 datasets shown in table 3.1 typify the variety found in (Hastie, Tibshirani, and Friedman 2009).

They span scientific, clinical, commercial and media fields. Note that they include many patches and pathways of everyday life – speaking, seeing, writing and reading – as well as specific scientific objects of knowledge such as galaxies, cancer, climate and national economies. This mixture is not unusual for machine learners. To give another example, the statistician Leo Breiman's 2001 article on random forests (Breiman 2001b), perhaps the most cited journal paper in the machine learning literature, displays a similarly diagonal line across human and non-human worlds:

> Glass, Breast cancer, Diabetes, Sonar, Vowel, Ionosphere, Vehicle, Soybean, German credit, Image, Ecoli, Votes, Liver, Letters, Satimages, Zip-code, Waveform, Twonorm, Threenorm, Ringnorm, (12).

In some ways, the improbable conjunction of spam email and cancer detection in machine learning continues what statistics as a field has always done: rove across scattered fields ranging from astronomy to statecraft, from zoology to epidemiology, gleaning data as it goes (see (Stigler 1986; Hacking 1990) for samples of itineraries).

In *Elements of Statistical Learning* and the field of machine learning more generally, something more is moving through and coordinating these datasets. The coordination might be rhetorical: the colligation of datasets – vowels, ozone, bone density, marketing, prostate cancer, and spam – in all their diversity suggests the mobility of machine learners. The combination of datasets deriving from network media, from medicine, from business administration and from cutting-edge life science (c.2000) suggests a tremendous, indeed almost spectacular miscibility, one that in principle could surprise us because there is otherwise little mixing between the settings and knowledge domains these datasets come from. How is this mixing, conformation

and homogenisation being done? The repetition of data-sets, the juxtaposition of discontinuous domains, and forms of movement construct and order continuities in the service of various forms of predictive and inferential knowledge. The miscible juxtapositions we encounter in machine learning enter into, it seems, a *regularity* or a common space, a space that displays strong tendencies to expand, accumulate and archive relations. Rather than peripatetic learning, the accumulation of diverse datasets attests to a prior ordering of data to afford its traversal.

The practices of naming and ordering, the sorting of different data types, the addressing and expansion of data exemplified in these diverse datasets can tell us a lot about how machine learning learns from data. The shapes, compositions and loci formed from the datasets enable the functioning of machine learners as they operate to generate statements, classifications and decisions. If machine learning can be understood as a constantly evolving diagram of practical abstractions, the way it draws on and relates to different forms of data matter. As we will see, machine learners transpose data in an increasingly extensive, heavily coordinated space, the vector space.

Truth is no longer in the table?

'This book is about learning from data' write Hastie, Tibshirani and Friedman on the first page of *Elements of Statistical Learning*, as they rapidly begin to iterate through some datasets. On the second page of the book, a table of spam email word frequencies appears (and the problem of spam classification is canonical in the machine learning literature – we return to in chapter 5). They come from the dataset `spam` (Cranor and LaMacchia 1998). On the third page, a complicated data graphic appears (Figure 1.1, (Hastie, Tibshirani, and Friedman

2009, 3). It is a scatterplot matrix of the `prostate` dataset included in the `R` package `ElemStatLearn`, the companion `R` package for the book. In a third example, a set of scanned handwritten numbers appears. These scans are images of zipcode or postcode numbers written on postal envelopes taken from the dataset `zip` (LeCun and Cortes 2012), and they differ from both the `spam` table and `prostate` plots because they directly resemble something in the world, which, however, happens to be numbers, and is, therefore, probably already recruited into data-making and data-circulating processes (a dataset we return to in chapter 8). The final example in the introduction, 'Example 4: DNA Expression Microarrays,' draws from biology, and particularly high-throughput genomic biology, a science that produces large amounts of data about biological processes by running many tests, or by constructing devices that generate many measurements, in this case, a DNA microarray.[3]

The table or the row-column addressable grid is common to all of these datasets. And yet, as we are about to see, machine learning in many ways deals with the collapse or liquidation of tabular datasets. 'Things, in their fundamental truth,' writes Foucault in *The Order of Things* 'have now escaped from the space of the table' (Foucault 1992 [1966], 239). Foucault writes in these pages about the fabled emergence of life, labour and language as the anchoring vertexes of a new triangle of knowledge and power structuring the figure of the 'human' in the 19th century.

Before the emergence of the characteristic sciences of the human — political economy, linguistics and biology — knowledges such as natural history, the general grammars, and philosophies of wealth (such as Adam Smith's work) had ordered empirical materials of diverse provenance in tables or grids. While the history of tables as

3. Some genomic data will be the focus of a later chapter 7. Machine learning during the 1990s and 2000s was in some ways boosted heavily by the advent of genomic biology with its large, enterprise style knowledge endeavours such as the Human Genome Project.

data forms reaches a long way back (see (Marchese 2013) for a broad historical overview that reaches back to Mesopotamia), Foucault argues that the Classical age first developed the system of grids that permitted ranking, sorting and ordering in tables. These grids replace the Renaissance tabulations based on 'buried similitudes' and 'invisible analogies' (Foucault 1992 [1966], 26). In pre-Classical tables, an image or figure from myth might lie alongside a measurement or a count of occurrences, and this proximity was ordered by systems of analogical association that spanned what we might today, in the wake of the 19th century, might see as incongruous (for instance, associations between medicine and Biblical prophecy).

The Classical Age grid or table, by contrast, brought plural and diverse resemblances into exhaustive systematic visible enumeration. As Foucault puts it:

> The space of order, which served as a *common place* for representation and for things, for empirical visibility and for the essential rules, which united the regularities of nature and the resemblances of imagination in the grid of identities and differences, which displayed the empirical sequence of representations in a simultaneous table, and made it possible to scan step by step, in accordance with a logical sequence, the totality of nature's elements thus rendered contemporaneous with one another (Foucault 1972, 239)

The table as space of order did not stand in isolation. It served a localized epistemic function in conjunction with other of knowledge such as experiment and mathematical proof. Since algebra or experimental *mathesis* only applied to 'simple natures' (planets in movement, dynamics of falling bodies, etc.), table-based knowledges such as taxonomy dealt with more complex natures. In the tables, systems of signs — for instance, the groupings established by the eighteenth century taxonomist Carl Linnaeus — sought to reduce complex

natures (plants, animals, etc.) to simpler forms as columns and rows in a table based on resemblances and similarities. Importantly, the table as space of order was a space of imagination in that one could begin to see continuities and differences between things (organisms, words, nations) by carefully ordering and scanning the table. 'Hedged in by calculus and genesis,' Foucault suggest, 'we have the area of the *table*' (Foucault 1992 [1966], 73). Note in passing that calculus and calculations bound the table only in relation to 'simple natures' whose identity and difference can be understood in the form of movements, rates and change in position.[4] This is an important limitation since it is precisely the complex natures in genesis that machine learning tries to engage using algebra, calculus, statistics and computation.

Finally (at least for our purposes), in the nineteenth century, a different form of ordering shattered tabulation based on enumerated similarity and ordered resemblance. Foucault figures this change as shattering the table into shards of order:

> this space of order is from now on shattered: there will be things, with their own organic structures, their hidden veins, the spaces that articulates them, the time that produces them; and then representation, a purely temporal succession, in which those things address themselves (always partially) to a subjectivity (Foucault 1992 [1966], 239-240).

Life, labour and language — Foucault's famous historically emergent triadic figure of the human — replace the enumerative, synoptic classificatory tables of the Classical age. Tables still abound in newer temporal, genetic orderings (almost any episode from the history of nineteenth and twentieth century statistics will confirm that; see (Stigler 1986, 2002)), but from now on tables are localized, relating to a place and changing in time, and functioning as shards of representational order addressed to (and often constitutive) of

4. In *Surveiller et Punir* or *Discipline and Punish* (Foucault 1977), Foucault returned to the question of the table in a slightly different context: an account of the operation of power in disciplinary institutions and knowledges. In these knowledges, the table becomes generative of 'as many effects as possible'.

subject position. A double interiority takes over. Things such as a language, a species or an economic system have their own genesis, ~~their own~~ temporality and historical existence. Our knowledges and indeed experience of them also become finite, historical, with their own dynamics and internal life. In this change, the table ~~itself~~ is no longer the foundation or distillation of knowledge. It is one knowledge apparatus amongst many. Tables, we might say, become merely data, inscriptions ~~pendant~~ on hidden structures and their genesis.

This brief résumé of a thread of argument in *The Order of Things* might help us reassess how machine learning goes into the data. The tables of `spam`, `prostate`, `image` and `microarray` data in contemporary machine learning do not operate in the way Linnaeus would have tabulated a table of living things based on similarities and resemblances or a Renaissance medical textbook might assemble analogical resemblances between disease and astronomy. As I will argue, measures of similarity and resemblance still operate strongly in machine learning as it moves through tables. In this respect, the Classical table and perhaps even the pre-Classical table returns with extended relevance. The repeated juxtaposition of tables of diverse provenance alongside each other, the incorporation of 'complex natures,' and the operational superimposition of different tables suggests machine learners still suggest synoptic alignments, but along different lines than the row-column grid of the Renaissance, Classical or even the later life-labour-language tables of the human sciences. Even if the tables in the opening pages of *Elements of Statistical Learning* concern work, life, language and economy and map ~~very~~ readily onto the anchor points, the new 'empiricities' (Foucault's term for the empirical problems), of labour, life and language or biopower that took root at this time (Foucault 1991), the ways machine learners

traverse them may not be recognisable in terms of these empiricities.

Instead, we are now confronted by kaleidoscopically transmuted tables whose expansion and open margins afford many formulations of similarity and difference. Already in the handwritten digits and the microarray data, scale and dimensions thwart tabular display of the data. In settings such as social media platforms or genomics in which machine learning operates, tables change rapidly in scale and sometimes in organisation. The multiplication and juxtaposition of tables suggests that we might be seeing the advent of a post-order space for regularities and resemblances, for simple and complex natures, encompassing knowledges ranging from humanities to traffic engineering.

The epistopic fault line in tables

The `ElemStatLearn` `R` package brings, as we have seen in the previous chapter, with it around 20 different datasets, including the four mentioned in the introduction to *Elements of Statistical Learning*. On the one hand, every data table indexes a localised complex of activities (clinical research, social media platform, financial transactions, etc.) with possible referential importance. On the other hand, for machine learners, the table is a space of potential similarities and differences both internal to the table ~~itself~~ (e.g. how much does row number 1000 differ from row 1,000,000) and associated with other tables (e.g. how much does this table of clinical test relate to that table of microarray data?). These internal and external differences entwine with each other in ways that create a fault line, an unstable yet generative line of diagonal movement. ~~It is this~~ fault or fold line, I propose, ~~that~~ diagrammatically distributes data tables into the expansive and moving substrata of the vector space.

The table has been vectorised. We might say that the vectorization of the table is epistopic. The term 'epistopic' comes from the work of the science studies scholar Mike Lynch whose account of scientific practice is particularly focused on ordinariness. Akin to what Foucault in the *Archaeology of Knowledge* terms a threshold of epistemologization (Foucault 1972, 195) Lynch characterizes the 'epistopic' as connecting localized practices ('topics') with 'familiar themes from epistemology and general methodology' in the local achievement of coherence in knowledge (Lynch 1993, 280). In other words, as the term itself suggests, an epistopic connects general epistemic themes such as validity, precision, specificity, error, confidence, expectation, likelihood, uncertainty, or approximation with a place, a 'local complex of activities' (281). This emphasis on epistemic location frames the problem of how the 'local complex' of a specific dataset encounters a generalizing epistemic practice such as machine learning.

Surface and depths: the problem of volume in data

The local complex of activities for associating datasets shatters tables from a different direction than that described by Foucault in *The Order of Things*. Algebra, linear algebra in particular, organizes and distributes differences in vector space. *Mathesis* in the form of algebraic operations of addition and multiplication of collections of tabular elements such as rows and columns, now re-defined as vectors, re-structure tabular data as a vector space, as a 'set' whose membership is only limited by the applicability of the constructing relations. These operations absorb and subtend differences in quality, type, kind and quantity.

Of the three example datasets (`prostate`, `spam` and `zip`), *Ele-*

	lcavol	lweight	age	lbph	svi	lcp	gleason	pgg45	lpsa	train
1	-0.58	2.77	50	-1.39	0	-1.39	6	0	-0.43	TRUE
2	-0.99	3.32	58	-1.39	0	-1.39	6	0	-0.16	TRUE
3	-0.51	2.69	74	-1.39	0	-1.39	7	20	-0.16	TRUE

Table 3.2: First rows of the ‘prostate‘ dataset

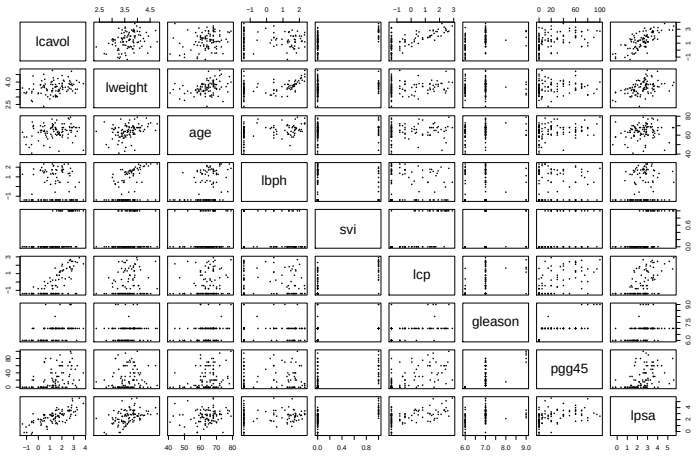

Figure 3.1: Scatter plot matrix of `prostate` data

ments of Statistical Learning returns most frequently to `prostate`. This dataset derives from the work of urologists working at Stanford (Stamey et al. 1989), and concerns various clinical measurements performed on men who were about to undergo radical prostatectomy. The measurements range across the volume and weight of the prostate, as well as levels of various prostate-related biomarkers such as PSA — prostate specific antigen. Several rows from the dataset are shown in Table 3.2. The first pages of the book had already exhaustively plotted all the variables in the dataset against each other using the table-related form of a scatter plot matrix (Hastie, Tibshirani, and Friedman 2009, 3) (shown in figure 3.1), and they return to the same data on almost a dozen occasions in the course of the book, subjecting it to repeated vectorization.

The contrast between the Table 3.2 and the Figure 3.1 already depends on a transformation intrinsic to vectorization. On the one hand, the table arrays all different data types in rows and columns. In the table, the relation between the different data types (the log of

the weight of prostate - `lwp` and `age` ~~for instance~~) is quite hard to see. Moreover, different kinds of variables stand side by side. `svi`, short for ~~'~~seminal vesicle invasion~~'~~ is a categorical variable. It takes the values ~~'~~true~~'~~ or ~~'~~false,~~'~~ shown here as `1` or `0`, but the other variables either measure or count things (years, sizes, or levels of antigens).

On the other hand, the scatter plot matrix also takes the form of a grid-like figure, but the cells of the grid are not occupied by numbers but by `x-y` plots of different pairs of variables in the `prostate` dataset. The ~~'~~matrix~~'~~ of figures shows of 72 plots is mirrored across the diagonal that runs from the ~~top-life~~ to bottom right in figure 3.1. Taking this folding into account, we see 36 unique plots with different data in each one. Each sub-plot displays the relation between two variables in the dataset as a scatter plot. Some variables such as `svi` are not ~~very~~ amenable to plotting in this way. More importantly perhaps, certain combinations of variables appear as flat loci that can be read as signs of relations between different variables. The scatter plot matrix constructs a tabular space in which relational contrasts between pairs of variables start to appear. In the light of these contrasts (and I use ~~'~~light~~'~~ here in an almost literal sense to refer to the way in which the architecture of the figure creates a space in which light scatters in varying patterns), the `prostate` dataset begins to expose relations that might be worth knowing about. We have moved on from the bare table of the dataset to a transformed tabulation, from a textual-numerical grid to a geometrical-numerical grid. Everything remains on the surface of a grid here, but the grid permits differences in relationality to begin to appear.

All of this somewhat precedes the operational formation of machine learning. Similar tables and plots are part and parcel of statistical data exploration more generally (see ~~(~~Beniger and Robyn 1978~~)~~ for

an historical account of quantitative graphics in statistics). The scatterplot matrix does not exhaust differences or relationality in the `prostate` dataset, but highlights a tendency to approach it from different angles (12 times in the *Elements of Statistical Learning*) in order to map the multiple relations or influences that remain opaque to even the most exhaustive matrices of plots. The scatterplot matrix shows pairs of variables in relation. If the crucial diagnostic factor in this case is an elevated level of the PSA (prostate specific antigen), how do we know what combinations of other measurements might be associated with its elevation? What if multiple variables affect the level of PSA?[5] This question can just about be pursued by scanning the matrix of plots, but not very stably since different data analysts might see different associations combining with each other there. Different statements or epistopics could be supported by the same figure.

The very question of relation between multiple variables and the predicted levels of PSA suggests the existence of a hidden, occluded or internal space that cannot be seen in a data table, and that cannot be brought to light even in the more complex geometry of a plot. This volume contains the locus of multiple relations, a locus inhering in a higher dimensional space, in this case, the nine dimensional space subtended by treating each of the nine variables or columns in the `prostate` dataset as occupying its own dimension. A different basis of order — the vector space — begins to take shape when dataset variables (usually columns in a table) become dimensions.[6]

Vector space expansion

To show how this space opens up, we might follow what happens to just one or two columns of the `prostate` data in the vector space

5. Despite the intensive work that Hastie and co-authors conduct on the `prostate` data, all with a view to better predicting PSA levels using volumes and weights of prostates, etc., Stamey and other urologists more than a decade or so concluded that PSA is not a good biomarker for prostate cancer. Stamey writes in 2004:

What is urgently needed is a serum marker for prostate cancer that is truly proportional to the volume and grade of this ubiquitous cancer, and solid observations on who should and should not be treated which will surely require randomized trials once such a marker is available. Since there is no such marker for any other organ confined cancer, little is likely to change the current state of overdiagnosis (and over-treatment) of prostate cancer, a cancer we all get if we live long enough. (Stamey et al. 2004, 1301)

6. Every distinct column in a table practically adds a new dimension to the vector space. Since the 1950s, problems of classification and prediction in high-dimensional spaces have been the object of mathematical interest. The mathematician Richard Bellman coined the term 'the curse of dimensionality' to describe how partitioning becomes more unstable as the dimensions of the space increase (Bellman 1961). The problem is that while the volume of a space increases exponentially with dimensions, the number of data points (actual measurements or observations) usually does not usually increase at the same rate. In high dimensional spaces, the data becomes more thinly spread out. It is hard to partitions

as it is vectorized. In the `prostate` dataset, some variables are continuous quantitative values, some are categorical (they represent membership in a group or category) and some are ordinal variables (they represent a ranking or order). How can different data types be located in vector space? ~~In order to~~ put classifications or categories into vector space, they need to be translated into the same *basis* as the quantitative variables with their rather more obvious geometrical and linear coordinate values. How does one geometrically or indeed algebraically render a category or ~~a~~ qualitative difference? The problem is solved via an expansion of the vector space through a form of binary coding that generates a new variable and hence a new dimension for each category:

> Qualitative variables are typically represented numerically by codes. The easiest case is when there are only two classes or categories, such as "success" or "failure," "survived" or "died." These are often represented by a single binary digit or bit as 0 or 1, or else by 1 and 1 … When there are more than two categories, several alternatives are available. The most useful and commonly used coding is via dummy variables. Here a K-level qualitative variable is represented by a vector of K binary variables or bits, only one of which is "on" at a time (Hastie, Tibshirani, and Friedman 2009, 12)

Again, the details are not so important here as the transformations that the vector space accommodates. A single qualitative or categorical variable expands into 'a vector of K binary variables or bits.' Qualitative data, once coded in this way, can be multiplied, added, and in short, handled algebraically using the same aggregate operations applied to numerical or continuous variables. Not only has the vector space expanded here, its expansion smooths over important ~~fault-lines~~ of difference that vertically divided the tabular data. Complex natures become simple natures. The different kinds of variables

qualitative and quantitative, discrete and continuous, nominal and ordinal can be accommodated by adding dimensions to the vector space. As Whitehead says, 'all things are vectors' (Whitehead 1960, 309).

Adding dimensions to vector space subsumes differences, but makes seeing the geometrically regular loci lines, planes, smooth surfaces in data distributed in this space more challenging. The many transformations in `prostate` that ensue in *Elements of Statistical Learning* become the locus of machine learning. In a historically significant transfiguration of the table, these expansions and we will see others, including *de novo* creations of constructed dimensions subtend differences in a vector space comprising elements defined purely by coordinate position and vectoral (having direction and extent) movement. Once this hidden, expandable and transformable (by rotation, displacement, or scaling) distribution of elements in space exists, strenuous efforts will be made to bring loci to light. Machine learners search for these loci or or feel for, to use Whitehead's term, along different lines. Sometimes a machine learner prehends vector space as filled with constantly varying proximities. It gathers and orders these proximities (for instance, as in the k nearest neighbours model) or in unsupervised methods such as k-means clustering (Hastie, Tibshirani, and Friedman 2009, 513). More commonly, machine learning draws lines or flat surfaces that constrain the volume.

The importance of lines and flat surfaces can hardly be underestimated in machine learning. Finding lines of best fit underpins many of the machine learners that attract more attention (neural nets, support vector machines, random forests). Linear regression with its pursuit of the straight line or plane projects the basic alignments of vector space. It renders all differences as distances and directions

of movement. Drawing lines or flat surfaces at various angles and directions is perhaps the main way in which the volume of data is traversed, and a relation between input and output, between predictors and prediction, consolidated as a loci or data strain loci or data strain.[7] The line of best fit has a ready generalization to higher dimensions, and a line can be diagrammed in the equations of linear algebra, the field of mathematics that operates on lines in spaces of arbitrary dimensions. Linear algebra operations exist for finding intersections between lines and planes, for manipulating collections of elements and aggregate forms such matrices through mappings and transformations (rotations, displacements or translations, skewing, and scaling), and above all, handling whole vector spaces as operational sets. It brings with it a set of formalisations — vector space, dimension, matrix, determinant, coordinate system, linear independence, eigenvectors and eigenvalues, inner-product space, etc. — that machine learners constantly and implicitly resort to invoke.[8]

Many of these operations quickly become difficult to geometrically figure.[9] Let us return to the equations for linear regression models (remembering that both C.S. Pierce and Andrew Ng advocate returning often to equations). The 'mainstay of statistics,' the linear regression model, usually appears diagrammatically in a more or less algebraic form:

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j \tag{3.1}$$

$$\hat{Y} = X_T \hat{\beta} \tag{3.2}$$

Equations 3.1 and 3.2 express a plane (or hyperplane) in increasingly diagrammatic abstraction. The possibility of diagramming a high dimensional space derives largely from linear algebra. Reading

7. One sign of the centrality of the line in machine learning can be seen, for instance, from the contents page of the book (Hastie, Tibshirani, and Friedman 2009, xiii-xxii). After the introduction of the linear model in the first chapter and its initial exposition in chapter 2 ('overview of supervised learning'), it forms the central topics of chapter 3 ('linear methods for regression'), chapter 4 ('linear methods for classification'), chapter 5 ('basis expansions and regularization'), chapter 6 ('kernel smoothing methods'), much of chapter 7 ('model assessment and selection'), chapter 8 ('model inference and averaging'), major parts of chapter 9 ('additive models, trees and related methods'), important parts of chapter 11 ('neural networks' — neural networks can be understood as a kind of regression model), the anchoring point of chapter 12 ('support vector machines and flexible discriminants') and the main focus in the final chapter ('high dimensional problems'). A similar topic distribution can be found in Andrew Ng's Cs229 lectures on machine learning. More than half of the 20 lectures concern linear models and their variants. See (*Lecture 13 | Machine Learning (Stanford)* 2008; *Lecture 6 | Machine Learning (Stanford)* 2008; *Lecture 7 | Machine Learning (Stanford)* 2008; *Lecture 10 | Machine Learning (Stanford)* 2008).

8. Along with statistics and probability, linear algebra is a such an important part of machine learning that many books and courses recommend students complete a linear algebra course before they study machine learning. Cathy O'Neill and Rachel Schutt advise: When you're developing your skill set as a data scientist, certain foundational pieces need to be in place first—statistics, linear algebra, some programming (Schutt and O'Neil 2013, 17).

9. We might add also approach the epistopic fault line in machine learning topologically. Over a decade ago, the cultural theorist Brian Massumi wrote that 'the space of experience is really, literally, physically a topological hyperspace of transformation' (Massumi 2002, 184). Much earlier, Gilles Deleuze had conceptualised Michel Foucault's philosophy as a topology, or 'thought of the outside' (Deleuze 1988b), as a set of movements that sought to map the diagrams that generated a 'kind of reality, a new model of truth' (35). More recently, this topological thinking has been extended and developed by Celia Lury amongst

Equation 3.1 from left to right, the expression $\hat{Y}$ already points to a set of calculated, predicted values, or a vector of y values, such as all the `lpsa` or PSA readings included in the `prostate` dataset. Similarly, the term X_j points to the table of all the other variables in the `prostate` dataset. Since there are 8 other variables, and close to 100 rows, X is a *matrix* — a higher dimensional table — of values, addressable by coordinates. Finally β_j are the pivotal coefficients or multiplying quantities that determine the slope or direction of the lines drawn. The second expression Equation 3.2 relies more fully on linear algebra. This is the linear model written in 'vector form' (Hastie, Tibshirani, and Friedman 2009, 11), or vectorized. The right hand side comprises two operations X^T, the transpose or rotation of the data, and implicitly — multiplication is hardly ever shown, but diagrammed by putting terms alongside each other — an *inner product* of the X matrix and the β parameters (to use model talk) or coefficients (to use linear algebra talk).[10]

The vector form in equation 3.1 diagrams an inclined plane that cannot be fully drawn in any figure, only projected perspectivally onto the surface of a graphic plot. While that line can never fully come to light, the diagrammatics of equations 3.1 and 3.2 express a way of constructing it and orienting it in vector space. Such expressions are epistopic in that they connect the local complex of activities indexed as tabulated data together through the diagonal diagrammatic element of a line or plane angling through vector space.

10. Carl Friedrich Gauss and Adrien-Marie Legendre's work on linear regression at this time is well-known. The first independent use of linear regression was Gauss' prediction of the location of an 'occluded volume,' the position of the asteroid Ceres after it reappeared in its orbit behind the sun. (Stigler 2002) — TBA page ref

Drawing lines in a common space of transformation

Once data is distributed in vector space, machine learners operate as transformations of that space into other vector spaces, the flat loci. Indeed from the perspective of vector space, machine learners

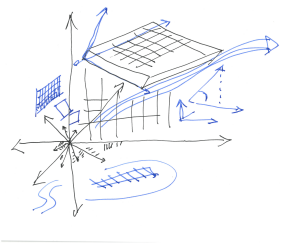

Figure 3.2: Vector space comprises transformations

are simple transformations or operations that map vector spaces into different ones, usually of lower but sometimes of higher dimensionality. For instance, 'drawing' the line of best fit through the `prostate` data or 'fitting a line' can be understood as a purely algebraic operation (although in practice, most machine learners are not purely algebraic—they optimise and probabilise, as we will see). Viewed in terms of linear algebra, the analytical or 'closed form solution' for the parameters of the linear model is given in equation 3.3:

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-}1\mathbf{X}^T\mathbf{y} \tag{3.3}$$

In this expression, linear algebraic operations on the data shown as $\mathbf{X}$ calculate the coefficients $\hat{\beta}$ that orient a plane cutting through the vector space.[11] The derivation of the analytical 'ordinary least squares' solution relies on some differential calculus as well as a range of linear algebra operations such as matrix transpose, inner product and matrix inversion, the details of which need not trouble us here. The relevant point is that equation 3.3 constructs a plane—a new vector—that traverses the density-shape of a dataset in its full dimensional vector space (nine dimensions in the case of `prostate`).[12]

Implicit vectorization in code and infrastructures

Vectorised transformations of data lie are the moving substrate of machine learning as it expands, but they are largely taken for granted

11. Perhaps more importantly, the linear algebraic expression of these operations presupposes that all the data, both the values used to build the model and the predicted values the model may generate as it is refined or put into operation somewhere, are contained in a common space, the vector space, a space whose formation and transformation can be progressively ramified and reiterated by various lines that either separate volumes in the space, or head in a direction that brings along most of the data. Not all of these lines are bound to be straight, and much of the variety and dispersion visible in machine learning techniques comes from efforts to construct different kinds of lines or different kinds of 'decision boundaries' (in the case of classification problems) in vector space (for instance, the k-nearest neighbors method does not construct straight lines, but somewhat meandering curves that weave between nearby vectors in the vector space; see (Hastie, Tibshirani, and Friedman 2009, 14-16)). Whether they are straight or not, the epistopic aspect of these lines remains prominent. Typically, many different statistical tests (Z-scores or standard errors,

as given space or commonsense ground. R coding practice instantiates vectorization in multiple ways, and is sometimes described as a 'vectorised programming language.' The vector space appears and operates just as directly in other programming languages designed for data practice (Octave, Matlab, Python's NumPy, or C++ Armadillo).

In vectorised languages such as R, transformations of a data structure expressed in one line of code simultaneously affect all the elements of the data structure. As the widely used *R Cookbook* puts it, 'many functions [in R] operate on entire vectors … and return a vector result' (Teetor 2011, 38). Or as *The Art of R Programming: A Tour of Statistical Software Design* by Norman Matloff puts it, 'the fundamental data type in R is the *vector*' (Matloff 2011, 24), and indeed in R, all data is vector. There are no individual data types, only varieties of vectors in R. There are many vectorised operations in the R core language and many to be found in packages (the popular 'plyr' package; vectorised operations can also be found in recent Python data analysis libraries such as `numpy` or `pandas` (McKinney 2012)). The fact that many of these vectorised operations occur implicitly suggests how pervasive vector space has become in data practice.[13]

Listing 3.1: Vectorising code

```
vector1 <- c(0,1,2,3,4,5,6,8,9,10)
vector2 <- c(0,1,2,3,4,5,6,8,9,10)


#procedural programming-style looped addition
result_looped = vector()
for (i in vector1) {
   result_looped[i] = vector1[i] + vector2[i]
}
result_looped


#vectorised addition
result_vectorised  <- vector1 + vector2
```

13. R sometimes presents difficulties for programmers trained to code using so-called procedural programming languages because it so thoroughly embraces the notion of the *vector*—and hence, regards all data as inhabiting vector space. In many mainstream programming languages, transformations of data rely on loops and array constructs in which some operation is successively repeated on each element of a data structure.

```
    result_vectorised
```

```
[1]  0  2  4  6  8 10 NA 16 18 20
```

```
[1]  0  2  4  6  8 10 12 16 18 20
```

The practical difference between the two approaches to moving through data is illustrated in the code listing 3.1 in which two 'vectors' of numbers are added together, in the first case using a classic `for`-loop construct, and in the second case using an implicitly vectorised arithmetic operation `+`. The difference between adding 1...10 using a loop or vector arithmetic is completely trivial here, or as we will soon see, nested operations are involved, these differences in coding significantly affect human-machine relations. This simultaneity is only apparent, since somehow the underlying code has to deal with all the individual elements, but vectorised programming languages take advantage of hardware optimisations or carefully-crafted low-level linear algebra libraries.[14] More importantly, this is a different mode of movement. Operations now longer step through a series of coordinates that address data elements, but wield planes, diagonals, cross-sections and whole-space transformations. Vectorized code reduces both data and computational frictions. The real stake in vectorizing data is not speed but transformation. It makes working with data less like iteration through data structures (lists, indexes, arrays, fields, dictionaries, variables), and more like folding a pliable material. Such practical shifts in feeling for data are mundane yet crucial to the epistopic movements in data.[15]

Vectorization also motivates increasingly parallel contemporary chip architectures, clusters of computers such as `hadoop` or `spark`, reallocation of computation to GPUs (Graphic Processing Unit), data-centre usage of FPGAs (Field Programmable Gate Arrays) and

14. Learning machine learning, and learning to implement machine learning techniques, is largely a matter of implementing series of matrix multiplications. As Andrew Ng advises his students,

> Almost any programming language you use will have great linear algebra libraries. And they will be high optimised to do that matrix-matrix multiplication very efficiently including taking advantage of any parallelism your computer. So that you can very efficiently make lots of predictions of lots of hypotheses (*Lecture 13 | Machine Learning (Stanford)* 2008, 10:50)

In other parts of his teaching, and indeed throughout the practice exercises and assignments, Ng stresses the value of implementing machine learning techniques for both understanding them and using them properly. But this is one case where implementation does not facilitate learning. Ng advises his learners against implementing their own matrix handling code. They should instead use the 'great linear algebra libraries' found in 'almost any programming language.' 'linear algebra libraries' multiply, transpose,

various other Cyclopean infrastructures of cloud computing. Many of these condensing and expanding movements of data are diagrammed in miniature in the R constructs as operators in vector space.

Lines traversing behind the light

How does the combination of algebraic vector space and vectorised code play out in data? 'We fit a linear model' write Hastie and co-authors, referring to one epistopic operation on `prostate` data in *Elements of Statistical Learning*. In R this might look like the code excerpt shown ~~below:~~

Listing 3.2: Building a `prostate` model

```
library(ElemStatLearn)
data(prostate)
columns_to_standardize = c(1,2,3,4,6,8,9)
prostate_standard = as.matrix(prostate[, columns_to_standardize
    ↪ ])
prostate_standard = as.data.frame(scale(prostate_standard))
prostate_standard = cbind(prostate_standard, gleason=prostate$
    ↪ gleason, svi = prostate$svi, train = prostate$train)
train = prostate$train ==TRUE
prostate_model = lm(lpsa~., prostate_standard[train,-10])
```

Estimate	Std. Error	t value	Pr(>|t|)
0.0227	1.1750	0.02	0.9847
0.5887	0.1097	5.37	0.0000
0.2279	0.0828	2.75	0.0079
-0.1226	0.0878	-1.40	0.1681
0.1821	0.0886	2.06	0.0443
-0.2499	0.1339	-1.87	0.0670
0.2313	0.1331	1.74	0.0875
-0.0256	0.1742	-0.15	0.8839
0.6386	0.2586	2.47	0.0165

Table 3.3: Fitting a linear model to the extttprostate dataset

Table 3.3 displays estimates of the coefficients or parameters $} that define the direction of a flat surface running through the vector space of the `prostate` data.[16] This new vector is a product of operations in the vectorized `prostate` data. Some vectorizing operations can be seen in R code in listing 3.2 (~~for instance~~, `as.matrix` or

16. From the epistopic viewpoint, the most obvious result of fitting a linear model is the production not of a line on a diagram or in a graphic. As we have seen, such lines cannot be easily rendered visible. Instead, the model generates a new column-vector of coefficients (see ~~T~~table 3.3) and some new numbers, *statistics*. This table is not as extensive as the original data, the $\mathbf{X}$ and Y vectors. But the names of the variables in the dataset appear as rows in the new table, a table that describes something of how

`scale(prostate_standard)`).

The "unique solution" to the problem of fitting a linear model to a given dataset using the popular method of "least squares" (Hastie, Tibshirani, and Friedman 2009, 12) is given by the operations we have seen in equation 3.3. This tightly coiled expression calculates the $\hat{\beta}$ parameters that set the slope and location of a flat surface or plane in nine-dimensional vector space using all of the `prostate` variables apart from one variable chosen as the response or predicted variable, in this case `lpsa`. X and y matrices are multiplied, transposed (a form of rotation that swaps rows for columns) and inverted (a more complex operation that finds another matrix) in a series of linear algebra transformations. Epitomising the implicitly vectorised code often seen in machine learning, calculating $\hat{\beta}$ for the `prostate` data only requires one line of `R` code:

Listing 3.3: ~~Closed form~~ evaluation of linear model parameters

```
beta_hat = ginv(t(X) %*% X) %*% t(X) %*% y
```

The implicit vectorization of the `R` code in the code listing 3.3, the fact that it already concretely operates in the vector space, operationalizes the concise diagrammaticism of equation 3.3 as a machine process. More importantly, the vectorised multiplication, transposition and inversion of data creates the new vector $\hat{\beta}$ whose variations can be explored, observed, graphed and varied in ways that go well beyond the statistical tests of significance, variation, and error reported in Table 3.3. (We will have occasion to return to these statistical estimates in chapter 5.) The play of values that starts to appear even in fitting one linear model will become much more significant when fitting hundreds or thousands of models, as some machine learners do.[17]

17. This ~~is an important~~ differentiation: it is not typical machine learning practice to construct one model, characterised by a single set of statistics (F scores, R^2 scores, t values, etc.). In practice, most machine learning techniques construct many models, and the efficacy of some predictive techniques de-

The vectorised table?

I started out from the observation that *Elements of Statistical Learning* mixes many datasets. The more abstract implications of vectorization and the forms of transformation movement it encourages and proliferates bring us back to the problem of how machine learning mixes datasets that span different settings. In short, vectorising computation makes the vector space, which we might understand as a resurgent form of the pre-Classical table, a table that tolerates ~~all~~ many ~~of~~ relations and similarities, operationally concrete and machinically abstract. It is no longer a visible diagram~~,~~ but a machinic process that multiplies and propagates into the world along many diagonal lines.

Machine learning relies on a broad but subtle transformation of data into vectors and a vector space. Slightly re~~-~~purposing Foucault's archaeology of tables in *The Order of Things*, I have suggested that vectorization remaps the grid of the table into the expanding dimensions of the vector space. This space accommodates both simple and complex natures. This is not the first such expansion of the table. We need only think of the relational database systems of the late 1960s~~,~~ and their multiplication of tables (Mackenzie 2012). But in the vectorised and matrix-form practices of the vector space, machine learning produces for the first time a meta(s)table volume that cannot be surfaced on a page or screen.

Does the vectorization of data lie a '~~-~~a long way from sense data' as Arendt suggest? In the diagrammatic operations of linear algebra on data~~,~~ and the vectorization of code, machine learning traverses dimensions that, as Arendt observes, cannot be immediately sensed. Whitehead's notion of data strain as 'a complex distribution of geometrical significance' suggests, however, that vectorization is

not a complete loss of feeling. Every machine learner inhabits and moves through the vector space along different strains. Sometimes their operations flatten the vector space down into lower dimensional sub-spaces as in the many 'dimensional reduction' machine learners such as principal component analysis, Latent Dirichlet Allocation or indeed the linear regression model that maps an irregular volume onto a plane. Sometimes they expand the vector space into a great many new dimensions or 'features' (as we saw with 'dummy variables' that embody categories, and as we will see with support vector machine classifiers in chapter 6 or the deep learners of chapter 8).

The epistopic transformation of datasets and tables into vector space reaches into and re-aligns communication and infrastructures. It acts as a powerful tensor on knowledges and operations of many different kinds as it transposes, inverts, and re-maps local complexes of activity. In following what happens to vectors, lists, matrices, arrays, dictionaries, sets, dataframes, series or tuples in data, we might get a sense of how the epistemic operations of predictive models, the supervised and unsupervised learners, the classifiers, the decision trees and the neural networks have purchase in data.

What is at stake in vectorizing data? It produces a common space that juxtaposes and mixes localized complex realities. The `prostate` dataset could be aggregated and melded as vectors with a microarray, heart disease or bone density datasets. In vector space, identities and differences change in nature. Similarity and belonging no longer rely on resemblance or a common genesis, but on measures of proximity or distance, on flat loci that run as vectors through the space. Vectorization, the deep saturation of the table by algebra, constitutes all relations as movements of transformation, diagonalization, inversion or rotation. The epistopic power of vectorization takes root in the ele-

mentary practices of machine learning, and engenders many variations amongst machine learners. Vectorisation also strains the production of knowledge through a loss of the visible geometry of tabular comparison. This loss of visibility is, as we will see, met by the production of new groups of statements, new visible forms and operational devices and infrastructures that accommodate the dimensional expansion of vector space. Infrastructural vectorization has often been called 'big data.'

The fascination of machine learning, its seemingly endless applications (I refer the reader back to the diagram of machine learning's vastness in chapter 2), owes much to the vector feeling, with its twin lures of ideal operationality — everything is a vector operation — and its tantalizing tendency to expand and to move. This feeling, the vector feeling, we might note, is not surprising. 'characteristically for Whitehead 'Feelings are "vectors"; for they feel what is there and transform it into what is here' (Whitehead 1960, 87).

Expansive data vectorization challenges contemporary critical to develop intuitions and value-relevant concepts describing vector-feelings or data strains. We lack good intuitions of how to do that partly because machine learning implicitly vectorizes its practice in code, in infrastructures and in highly condensed diagrammatic forms. My aim in undertaking an archaeology of the transformations of tables into vector spaces is to unwind or de-diagonalise some of the operations rippling through different treatments of data. The act of diagramming how machine learners vectorise data densities begins to locate and unravel the processes of knowing, predicting and deciding on which many aspects of the turn to data rely. The vectoral operations we have just been viewing are themselves organised and aligned by other lines of diagrammatic movement that shape surfaces

in more convoluted forms.

4

Machines finding functions

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

‘All knowledge,’ hypothesises Pedro Domingos, ‘past, present,and future can be derived from data by a single, universal learning algorithm’ (Domingos 2015, 25). How will the ‘single, universal’ algorithm learn, how will it ‘epistemologize,’ to use Foucault’s term, ‘our cultures’?

In practice, 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’, cautions ~~Domingos~~ readers of the *Communications of the ACM*, ‘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’ echoes 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

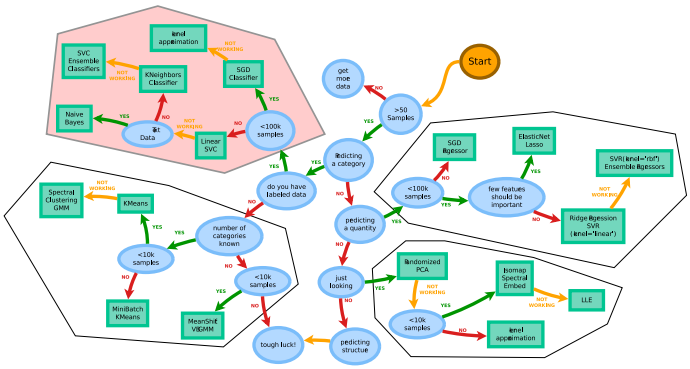

Figure 4.1: `scikit-learn` map of machine learning techniques

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 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 weed-like 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 machine learners, 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 dual effect of profuse accumulation and the ideal a single, universal machine learner arise and hold together?

The machine learners I have just cited present that profusion as a

problem of the piling up of techniques. 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 4.1 comes from a machine learning 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, or the many overlapping packages in `R`, code libraries such as `scikit-learn` 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]

The architecture of 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. These divisions, maps and classifications help order the techniques, but they obscure the process that first generates a competing profusion of machine learners.

If, as I have suggested earlier, we understand knowledge in terms of the radically re-conceptualised statements that Foucault described in *The Archaeology of Knowledge*, then statements comprise various

1. Similarly, for `R` code, the *Comprehensive R Archive Network* tabulates key libraries of `R` code in a machine learning 'task view' (Hothorn 2014).

units (sentences, series, tables, propositions, diagrams, equations, numbers) mapped to a field of objects, subject positions and domains of coordinations and reuse by an enunciative function (Foucault 1972, 106). Confronted by a profusion of machine learners and the idea of a single, universal machine learning, an archaeological analysis attends to the enunciative function that multiplies meanings and operations.

We might understand the enunciative function as the generative process that proliferates machine learners. The listing and mapping of accumulated techniques, whether in the form of textbooks such as *Elements of Statistical Learning* or a code library such as `scikit-learn`, together with the many attempts to unify them (Domingo's 'single, universal algorithm', `scikit-learn`'s map, *Elements of Statistical Learning*'s statistical theory) suggests a commonality in the production of statements. As I will argue in this chapter, there are so many techniques, algorithms and ways of deriving knowledge from data in machine learning because statements are actually rare in this operational formation. 'Because statements are rare,' writes Foucault, 'they are collected in unifying totalities, and the meanings to be found in them are multiplied' (120).

Learning functions

The rarity of statements amidst the profusion of machine learners revolve around a single operator, the function.[2] Table 4.1 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,

2. I discuss the sense of function as operation or process in chapter 7. 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.

dominate. Machine learners ~~'find', 'estimate,' 'approximate,' 'analyse' and sometimes 'decompose'~~ mathematical functions. The primary mathematical sense of a function refers to a ~~relation – a mapping~~ 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. For instance, ~~a~~ 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 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 include the logistic function (discussed ~~below~~), probability density functions (PDF) for different probability distributions (Gaussian, Bernoulli, Binomial, Beta, Gamma, ~~etc. See~~ chapter 5), and error, cost, loss or objective functions (these four are almost synonymous). (The latter group I discuss below because they underpin many claims that machine learners learn.)

As we will see, from the perspective of the function, machine learning can be understood as a function-finding operation. Implicitly or explicitly, machine learners find a mathematical ~~expression – a function – approximating~~ the social, technical, financial, transactional, biological, brain, heart or group process that flowed the data in question into vector space. Regardless of the application, no single mathematical function perfectly or uniques expresses data. Many if not infinite functions can approximate any given data. Even if

there was a master algorithm, therefore, it would be concerned with a field of functions, and it would entail observation, classification and selection (finding, in short) in deriving knowledge from data.

Supervised, unsupervised, reinforcement learning and functions

The capacity of machine learners to learn is very closely linked to forms of observation that co-produce knowledge. The optics of this observation of machine learners vary but they are always partial or incomplete, partly because of the dimensionality of vector space and partly because of the domains in which machine learning operates. While the field is pragmatic in its commitment to 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 distinguishes between three broadly different kinds of *learning* — supervised, unsupervised, and reinforcement — in terms of their observability. *Elements of Statistical Learning* presents the distinction between supervised and unsupervised learning:

> 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)

Supervised learning in general terms constructs a model by training it on some sample data (the training data), and then evaluating the model's effectiveness in classifying or predicting unseen test data whose actual values are already known. The 'clear measure of success' in relation to 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, machine learners observe how a function (or functions) changes as a model transforms, partitions or maps the data.

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

Viewed as enunciative function, machine learning makes statements through operations that treat functions [illegible]. At the same time, opacity — 'no direct measure of success' — is generative in machine learning. *Elements of Statistical Learning* admits that success cannot be measured and that this inaccessible difficulty has led to proliferating methods, transformations and changes. 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.

Which function operates?

Differences between machine learners can be described using mathematical functions. That is, machine learners operate as functions and observations of those operations also constitute functions. Functions instantiate *both* the operations and the ordering of those operations.

For instance, classifiers, or machine learners that classify, are often

identified directly with functions:

> A classifier … 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 $1, 2, \ldots, J$ (Breiman et al. 1984, 4).

Writing in the 1980s, the statistician Leo Breiman identifies classifiers — perhaps the key operational achievement of machine learning and certainly the catalyst of many applications — with 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 (for instance, `cat` or `not cat` in the case of `kittydar`).

The identification of machine learning with functions appears in the first pages of most machine learning textbooks. Viewed operationally, learning in machine learning means finding a function that can identify or predict patterns in the data. As *Elements of Statistical Learning* formulates it,

> 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).

This statement of learning compresses several layers in the function. It posits the existence of *the* function that generated the data as a foundation. This function figures as a ground truth existentially imputed to the world. It also refers to 'finding … $\hat{f}(x)$' where the'^'indicates a 'useful' approximation. A leading theorist of learning theory Vladimir Vapnik echoes the statement of learning as a function: '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 'function-fitting paradigm' as (Hastie, Tibshirani, and Friedman 2009, 29) terms it, emphasises this double layering of

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 7 discusses the support vector machine.

function as an observed approximation. Most importantly, *learning* here is understood as finding. Despite many 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. 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).

The superimposed or doubling of function as operation and observer is hardly ever explicitly mentioned by machine learners. The pages of (Hastie, Tibshirani, and Friedman 2009) present a series of '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 mixed list draws from a pool of several hundred mathematical functions commonly used in science and engineering.[5] Clearly neither machine learners or critical researchers can expect to understand the functioning of all these functions in any great detail. While this prickly list of terms begins confirms the salience of functions in machine learning (as perhaps in many other science and engineering disciplines), certain basic difference between functions might be a way to map the interplay of operational and observational functions. We can already see in this list that function

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.

are diverse. Sometimes, the function 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.'

What does a function learn?

We wish to know: in what sense does a machine learner learn? This question can now be re-framed: how to machine learners find functions? For critical thought, this is a vexing question, for if function-finding agency inheres in machines and devices, then the politics of human-machine relations, and the practices of knowledge production shift. The philosopher of science Isabelle Stengers sets tight limits on functions:

> 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)

If they cannot learn 'new habits,' what can functions learn? 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 implicit in their structure. Similar statements might be found in many philosophical writings on science and on mathematics in particular.[6] But throughout in her writing Stengers explicitly affirms *experimental practice*, much of which depends on functions and their operations (Stengers 2008). It might be better to say that she limits the agency of functions in isolation in order to highlight their specific power in science: 'celebrating the exceptional character of the experimental achievement very

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). 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.

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 experimentally constructed relays:

> 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, devices, facts (statements) and mathematical functions in a heterogeneous weave.[7] Given that learning is not radically innate to machines, it might ~~better~~ be understood as an experimental achievement. When a biomedical researcher ~~uses~~ seeks 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~~ might estimate and evaluate ~~their~~ predictions using classical statistical approaches (analysis of variance, correlations, regression analysis, etc.). The question from Stengers' standpoint is this: what happens to the structure of referrals through experiments and the existing knowledge when functions are said to learn? ~~In order to~~ address this question, we need to delineate how functions function in machine learning.

At first glance, machine learning as a field is not ~~very~~ experimental (even if it radically influences the conduct of experiments in many scientific fields; see chapter 7). It lacks the apparatus, ~~the~~ instru-

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

ments, the laboratories, field sites or clinics of experimental practice. Experimentation takes place principally in the form of rendering diagrammatically the relays or referrals between different functions as they traverse data. They appear in graphic forms as plots. The diagrammatic entanglement of operation and observation in functions is not surprising. The historical invention of the term 'function' and a notation for writing functions by the philosopher G.W. Leibniz in the 17th century pertained to the problem of describing continuous variations in curves. Functions for Leibniz describe variations in response to other changes (y may change in response to a change in x), but they can also describe tendencies in functions (as a derivative function describes the sensitivity or rate of change of the slope of curve). Identifying and locating important tendencies or changes in functions —*singularities* in curves- also preoccupies the function-finding done in machine learning. In contrast to the vector space that expands to accommodate all transformations, the many observational elements such as graphic objects stage observations of tendencies or change points. The experimental relay or referral, the power to confer on things the power to confer on the machine learner (human-machine) the power to speaker in their name (Stengers 2000, 89), pivots around the double layer of functions. An operational function transforms the vector space and an observational function generates statements concerning degrees and rates of success, fit or error.

While we have yet to see how a function can observe, we can readily see some of the effect of the coupling between operational and observational functions. In the several hundred colour graphic plots in *Elements of Statistical Learning*, a striking mixture of network diagrams, scatterplots, barcharts, histograms, heatmaps, boxplots, maps, contour plots, dendrograms and 3D plots exhibit different aspects

of this tension between operation and observation. Many of these graphic forms are common in statistics as statements of variation or tendency in data (histograms and boxplots). Others relate specifically to machine learning (~~for instance, ROC~~ Receiver Operating Curve – or regularization path plots). A significant proportion of these graphics do not display data from experiments or measurements, but diagram variations in the operational function that transforms the data in relation to some criteria of observations (~~for instance~~, prediction errors or purity of classification).

Observing with curves: the logistic function

How can a function observe? As we have already seen, machine learners often learn by '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 vector space cannot be fully seen and its operational transformation into lines, planes, or smooth surfaces often remain occluded. Graphic plots and statistical summaries offer perspectival views on those transformations, but machine learners observe many of those transformations by adding a feedback loop between the transformations (fitting a line, building a decision tree, adjusting the weights in a neural net, etc.) and observed outcome.

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). As operational functions in machine learning, they illustrate a transformation of discrete values into continuous values. As observational functions, they exemplify observability, as we will see, in the form of their differentiability. An example of a sigmoid function, the logistic