

Machine Learning: Archaeology of a Data Practice

Adrian Mackenzie

Preface

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

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

1. Typesetting code and names of devices and datasets in a san serif font like `this`;
2. Presenting formulae, functions, and equations using the bristling indexicality of mathematical typography

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

A dense feedback signal runs through the many propositions, formulations, diagrams, equations, citations and images in this book. I've been writing code for years (Mackenzie 2006). Writing code was nearly always something distant from writing about code. Recent developments in ways of analysing and publishing scientific data bring coding and writing closer together, such

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

The capacity to mix text, code and images depends on an ensemble of

programming languages! as mode of writing

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

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

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

`ipython` combines text and Python code more continuously, but at the cost of editing and writing code and text in a browser window. Most of the conveniences and affordances of text editing software is lost. While `ipython` focuses on interactive computation, `knitr` focuses on bringing together scientific document formatting and computation. From the perspective of praxiography, given that both can include code written in other languages (that is, python code can be processed by `knitr`, and R code executed in `ipython`), the differences are not crucially important [P3]. This whole book could have been written using just Python, since Python is a popular general purpose programming language, and many statistical, machine learning and data analysis libraries have been written for Python. Widely used Python modules such as NumPy, SciPy, Scikit-learn, open-cv or Pandas allow anything done in R to be done in Python. It is difficult to generalise about the differences between the two programming languages. I have used both, sometimes to highlight tensions between the somewhat more research-oriented R and the more practical applications typical of Python, and sometimes because code in one language is more easily understood than the other.

Acknowledgments

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

Decades later, from 2007-2012, I benefited greatly from a research position in the UK Economic and Social Research Council-funded Centre for Economic and Social Aspects of Genomics at Lancaster University. Certain colleagues there, initially in the Sociomics Core Facility, participated in the inception of this book. Ruth McNally first of all, but also Paul Oldham, Maureen McNeil, Richard Tutton, and Brian Wynne were participants in many discussions concerning the transformation of life sciences around which my interest in machine learning first crystallised. Various academic staff in the Department of Applied Mathematics and Statistics at Lancaster University shepherded me through their post-graduate training courses: Brian Francis for his course of 'Data Mining,' David Lucy for his course on 'Bayesian Statistics', Thomas Jakl for his course 'Genomic Data Analysis' and TBA's course on 'Missing Data.' My colleagues in science studies at Lancaster, especially Maggie Mort, Lucy Suchman, and Claire Waterton have . I have

been very fortunate to have worked with inspiring and adventurous doctoral students at Lancaster during the writing of this book. Lara Houston, Mette Kragh Furbo, Felipe Raglanti, Emils Kilis, Xaroula Charalampia, and Nina Ellis have all helped and indeed challenged me in different ways. Sjoerd Bollebakker very kindly updated many of the scientific literature searches towards the end of the book's writing.

Contents

Preface	ii
Acknowledgments	vii
List of Figures	xi
List of Tables	xiii
1 Introduction: Into the Data	1
All control to the machine learners?	9
The archaeology of a data practice	12
Massive asymmetries in a common world	15
What cannot be automated?	19
Different abstractions in machine learning?	22
The diagram	24
2 Diagramming machine learning	29
A diagram of the elements of machine learning	31
The obdurate mathematical glint of machine learning	38
CS229, 2007: returning again and again to certain features	43

The learning of machine learning	48
What the perceptron latches onto	51
Machine learning implemented as program	58
The diagram of a hyperobject	67
3 The vector space and its movements	71
Mixing media, medicine, business and biology	76
Truth is no longer in the table?	78
The epistopic fault line in tables	82
Surface and depths: the problem of volume in data	84
Vector spaces expand	89
Traversing behind the light	90
Drawing lines in a common space of transformation	95
-ply: vectorisation and the transformation of common space	97
'We fit a linear model'	101
The vectorised table?	103
4 Machines finding functions	109
Supervised or unsupervised, who learns what?	113
Which function operates?	116
What does a function learn?	119
Diagramming with curves: the logistic function	124
The cost of curves in machine learning	127
The cost of curves in machine learning	129

Cost, loss, objective and optimisation	133
Gradients as partial observers	136
5 <i>N = all: Probabilisation</i>	145
Machine learning as statistics inside out	149
Naive Bayes and the distribution of probabilities	156
Spam: when $\forall N$ is too much?	159
The improbable success of the Naive Bayes classifier	165
Ancestral probabilities in documents: inference and prediction	168
Statistical decompositions: high bias, low variance and observation of errors	171
Does machine learning construct a new statistical reality?	174
6 Patterns of accumulation and dispersion	179
Splitting or recursive binary partitioning and the growth of trees .	184
1984: Cutbacks on recursive partitioning	188
The successful dispersion of the support vector machine	198
The decision boundary blurs?	201
Bending the decision boundary	207
Patterns as institutional not spatial-temporal	212
7 Regularizing objects	217
The positivity of the genome	223
The advent of ‘wide, dirty and mixed’ data	229
Regularizing machine learning in genomics	235

The proliferation of discoveries	242
Variations in the object or in the machine learner: k nearest neighbours models	245
Whole genome functions	248
8 Propagating subject positions	255
Propagating neural nets in machine learning	260
A privileged machine and its diagrammatic forms	269
The subjects of a hidden operation	276
Algorithms that propagate errors	279
Competitions as examination	281
Subject positions in the confusion matrix	290
9 Conclusion: Out of the Data	295
Bibliography	301
Index	309

List of Figures

1.1	Google Trends search volume for ‘machine learning’ and related query terms in English, globally 2004-2015	3
1.2	Cat as histogram of gradients	5
1.3	Machine learners in scientific literature	22
2.1	Pages cited from <i>Elements of Statistical Learning</i>	36
2.2	Publications cited by <i>Elements of Statistical Learning</i>	37
2.3	Class notes lecture 5, Stanford CS229, 2007	45
2.4	1958 perceptron and 2001 neural net compared	49
3.1	Scatter plot matrix of prostate data	86
3.2	Common space in linear equations	95
4.1	‘scikit-learn’ map of machine learning techniques [TBA: ref to diagram]	110
4.3	Logistic or sigmoid function	125
4.4	South African Heart disease regularization plot	131
4.5	South African Heart disease decision plane	131
4.6	Gradient ascent for logistic regression	137

6.1	Recursive partitioning of the feature space	185
6.2	AID classifies annual earnings (Morgan & Sonquist, 1963, 430)	187
6.3	SVM on Anderson's iris dataset	200
6.4	MNIST Postal Digits	203
6.5	Margins in a support vector machine	205
7.1	A human genome diagrammed using the Circos	226
7.2	Hierarchical clustering of the SBRCT gene expression data .	234
7.3	Shrinkage path of coefficients for Lasso regression on (Golub,1999) leukemia data	242
7.4	The earliest formulation of the <i>k</i> nearest neighbours model from Evelyn Fix and Joseph Hodges' 1951 paper [@Fix1951]	247
7.5	KNN models for simulated data in two classes	247
8.1	An early publication of the back-propagation algorithm: Rumelhart, Hinton and William's 1985 paper [@Rumelhart_1985]	263
8.2	Neural network topology for 5-hidden unit 'titanic' data . .	275
8.3	Kaggle data science competitions	284

List of Tables

1.1	A small sample of titles of scientific articles that use machine learning in relation to "climate"	27
2.2	The Boolean function NOT-AND truth table	53
2.1	Subject categories of academic research literature citing <i>Elements of Statistical Learning</i> 2001-2015. These subject categories derive from Thomson-Reuter <i>Web of Science</i>	68
2.3	Iterative change in weights as a perceptron learns the NAND function	69
2.4	ELEMSTATLEARN suggest R packages	70
3.1	Datasets in <i>Elements of Statistical Learning</i>	106
3.2	First rows of the 'prostate' dataset	106
3.3	Fitting a linear model	107
4.1	Sample of highly cited machine learning publications referring to "function" in title or keyword	143
4.2	Sample of highly cited scientific publications referring to "logistic regression" in title or keyword	144

5.1 Some structuring differences in machine learning	151
6.1 References to Morgan and Sonquist's Automatic Interaction Detector	186
6.2 Most cited papers on support vector machines	215
7.1 First 5 rows of Fisher's 'iris' dataset	230
7.2 Small round blue-cell tumour data sample (Khan, 2001) . .	233
8.1 Techniques and concepts most frequently mentioned in ma- chine learning publication keywords, 1955-2015	293
8.2 The highest prize money machine learning competitions on Kaggle	294

Mitchell, Tom
machine
learner!computer
program as
Breiman, Leo

Chapter 1

Introduction: Into the Data

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

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

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

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

machine
learner!National
Security Agency
Skynet

data science!relation to
machine learning

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

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

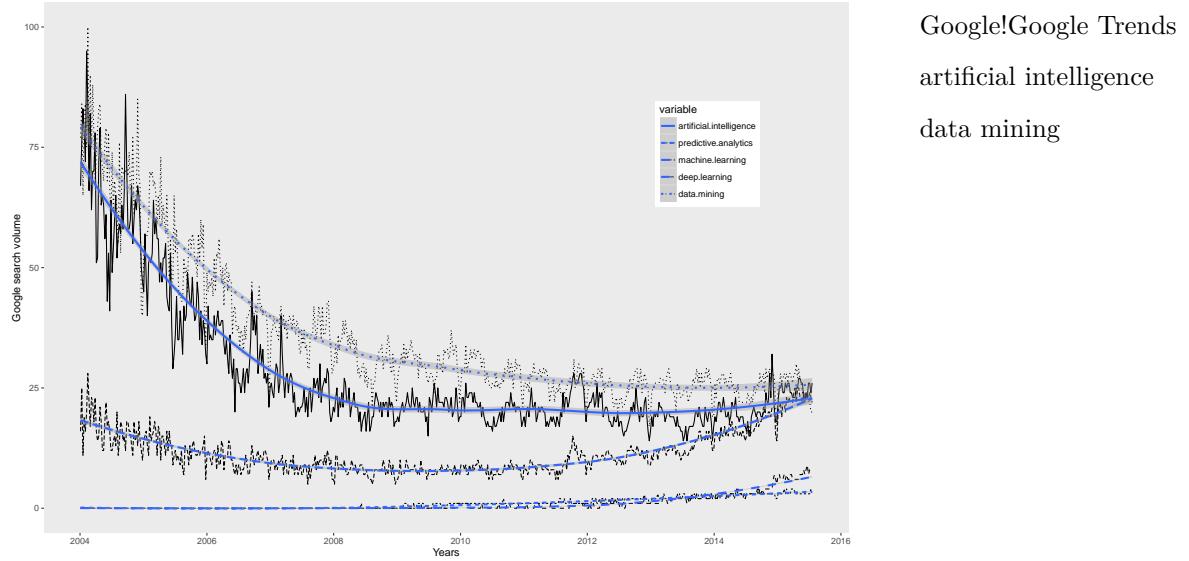


Figure 1.1: Google Trends search volume for ‘machine learning’ and related query terms in English, globally 2004-2015

decade. If we search for terms such as `artificial.intelligence`, `predictive.analytics`, `machine.learning`, `deep.learning`, `data.mining` on the [Google Trends](#) service , the results for the last decade or so suggest changing interest in these topics.

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

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

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

to specific invents such as significant press releases, public debates, media attention and film releases.

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

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

What does it mean that machine learners surface in so many different places, from fMRIs to Facebook, from fisheries management to Al Queda courier network monitoring? This book is an attempt to answer that question by describing the general horizon of machine learning as a data practice in its specificity. Take the case of **kittydar**, a machine learner in the area of image recognition (see [kittydar](#)): ‘Kittydar is short for kitty radar. Kittydar takes an image (canvas) and tells you the locations of all the cats in the image’ (H. Arthur 2012) . This playful piece of code demonstrates the

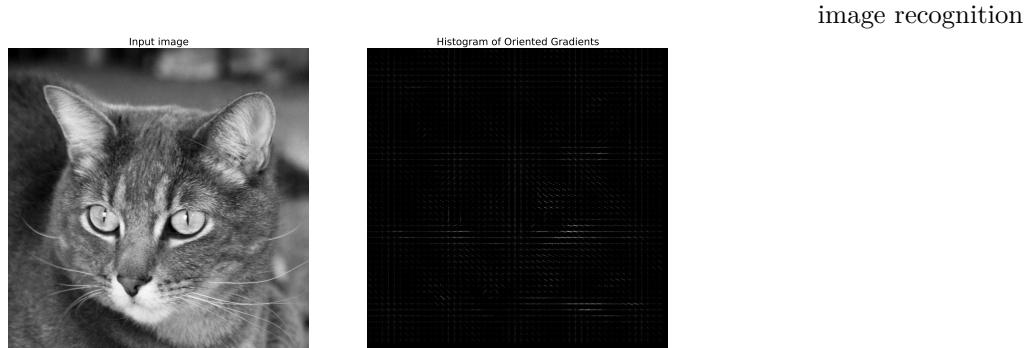


Figure 1.2: Close up of cat. The image on the left is already signal processed as JPEG format file. The image on the right is further signal processed using histogram of oriented gradients (HOG) edge detection. `kittydar` models HOG features. Photo courtesy photos-public-domain.com

deployment of a machine learner in the mundane, not to say banal, domain of cat photos on the internet. Heather Arthur, who developed `kittydar` writes:

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

classification	for the node training scripts (H. Arthur 2012).
neural	
network!seemachine	
learner!neural net	
support vector	This toy example of machine learning data practice finds cat heads in digital photographs. Based on how it locates cats, we can begin to imagine similar pattern recognition techniques in use in self-driving cars, border security systems, military robots or wherever there is something to be seen.
machine!seemachine	
learner!support vector	
machine	
linear regression	Kittydar runs in Javascript in a web browser, and only detects the presence of cats that face forward. As Arthur's description suggests, the software finds cats by cutting the images into smaller windows. For each window, it measures a set of gradients running from light and dark, and then compares these measurements to the gradients of known cat images (the so-called 'training dataset'). The device associates sudden shifts from light to dark with the regular features on cats' faces. The work of classification according to the simple categories of 'cat' or 'not cat' is given either to a neural network (as discussed in chapter ??), a typical machine learning technique and one that has recently been heavily developed by researchers at Google (Le et al. 2011), themselves working on images of cats among other things taken from Youtube videos (BBC 2012) , or to a support vector machine, a technique first developed in the 1990s by researchers working at IBM (see chapter 6).
support vector machine	
k-means clustering	
\	
nearest	
neighbours seemachine	
learner!_k_nearest	
neighbours	
principal component	
analysis	
machine learner!Naive	
Bayes	Machine learners range from the mundane to the esoteric, from the miniature to the infrastructural sublime of computational clouds. Like kittydar, they often classify, rank, cluster and estimate things. Their names accumulate and repeat in textbooks, instructional courses, website tutorials, software libraries and code listings: linear regression , logistic regression, neural networks , linear discriminant analysis , support vector machines, k-means clustering, decision trees decision tree!see{machine learner!decision tree}, k nearest neighbours, random forests, principal component analysis, or naive Bayes classifier to name just some of the most commonly used. Sometimes

they have proper names: RF-ACE, Le-Net5, or C4.5. These names refer to predictive models and to computational algorithms of various ilk and provenance. Dauntlessly intricate practices – normalization, regularization, cross-validation, feature engineering, feature selection, optimisation – suture datasets into shapes they can recognise. The techniques, algorithms and models are not necessarily startling new or novel. They take shape against a background of more than a century of work in mathematics, statistics, computer science as well as various scientific fields ranging from anthropology to zoology. Mathematical constructs drawn from linear algebra, differential calculus, numerical optimization and probability theory pervade practice in the field. [^0.15]

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

control
generalization
Foucault, Michel
operation, field of

disaggregate cancer sub-types, patterns of speciation in birds, identify Al-Qaeda operatives or attune the movements of a prosthetic hand to nerve activations in an amputees' residual limb. Power and knowledge, control and positivity mix thoroughly in machine learning. I find it difficult to separate the novel human-machine relations that have been taking shape in machine learners in their knowledge-making practices from the systems of control, classification, decision, power and discrimination they also operate. I suspect I'm not alone in that difficulty, and for that reason, I attempt in the chapters that follow to carefully describe practices, knowledges, forms and terrains specific to machine learning.

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

All control to the machine learners?

control!control
revolution

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

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

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

As Amoore writes, some potential futures are being ‘ruled out’ as these devices are put to work. Anna Munster puts the challenge more bluntly:

\ decision infrastructure handwriting recognition|seealsodigit recognition algorithm!primacy

'prediction takes down potential' (Munster 2013). I find much to agree with here. Machine learning is a convoluted but nevertheless concrete and historically specific form of calculation calculation|see{mathematics!calculation} (as we will see in exploring algebraic operations in chapter 3, in finding and optimising certain mathematical functions in chapter 4 or in characterising and shaping probability distributions in chapter 5). It tends to algorithmically mediate future-oriented decisions (although all too often, very near-future decisions). It automates, and in many cases, specifically aims to automate that which hitherto appeared impossible to automate. And this automation, with all the investment it attracts (in the form of professional lives, in the form of infrastructures , in research funding, in reorganisation of corporate and government processes, etc.) does rule out some and reinforce other futures. As for consequences, we need only consider some of the many forms of work that have already been affected by or soon could be affected by machine learning. Postal service clerks no longer sort the mail because neural net-based handwriting recognition reads addresses on envelopes . Locomotives, cars and trucks are already driven by machine learners, and soon driving may not be same occupational cultural it was. Hundreds of occupational categories have to some degree or other machine learners in their near future.²

Given the consequential weight of such algorithms and calculations, how do we uncover the moments that come together in them? We might see algorithms as making calculation automatic. In various scholarly and political debates around changes business, media, education, health, government or science, quasi-omnipotent agency has been imputed to algorithms [Pasquinelli (2014);Neyland (2014);Totaro and Ninno (2014);M. Smith (2013); Beer and Burrows (2013);Fuller and Goffey (2012);Wilf (2013); Baracas, Hood,

2. Carl Benedikt Frey and Michael Osborne model the chances of occupational change for 700 occupations using, aptly enough, the machine learning technique of Gaussian Processes (Frey_2013).

and Ziewitz (2013); Gillespie (2014); (Cheney-Lippold 2011; Alexander R Galloway 2004) or sometimes just ‘the algorithm.’ The power of algorithms in the social science and humanities literature is understood in different ways, but there is general agreement that algorithms are powerful, or at least, can bear down heavily on people’s lives and conduct, re-configuring, for instance, culture as algorithmic (Hallinan and Striphas 2014). In what does this power and increasing prestige consist? What imbues algorithms with power? Is there some general or ‘breakout’ feature such as recursivity) or abstraction that configures contemporary rationality algorithmically?

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

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

‘Moving at the same level of abstraction as the algorithm’ offers some purchase as a formulation, but I find ‘the’ algorithm, ‘the level of abstraction’ and ‘a mathematical way’ all troublesome as ways of qualifying machine learners. Which algorithm, what kind of abstraction and which mathematical way should we focus on? From the standpoint of diverse machine learners and the many different person-thing-object-relations they encompass, there is no single a-historical level of abstraction, but something more like a

algorithm!recursivity

abstraction!in

algorithms

Pasquinelli, Paolo

abstraction!algorithm as torque and flux of different moments of abstraction at work in generalizing, automation!limits of classifying, circulating and stratifying in the midst of transient and plural machine multiplicities. [^0.12]

learner!diagrammatic composition of mathematics!as stabilisation of practice

The archaeology of a data practice

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

If abstractions, mathematics or the algorithm loom large in classification practices today, how do we re-establish their connection to the workings of

power without pre-emptively ascribing potency to the mathematical way, to the algorithm or abstraction? In the chapters that follow, I map machine learning data practices in greater empirical and conceptual depth, and develop some terms to describe them less generically (common vector space in Chapter 3, partial observer trajectory in Chapter 4, decision surface in Chapter 6, inverse probabilization in Chapter 5), but all of the facets I describe cluster around the problem of understanding how an almost banal, not esoteric, operation is repeated, borrowed, copied and diffused in so many settings.

If we understand machine learning as a specific data practice, then the forms of order that result from it also become more tangible. Nearly all machine learners can classify things. They are often simply called ‘classifiers.’ *Kittydar* classifies images as cats, but categorisation and classification applies much more generally.³ Many machine learners predict categories, levels, rankings, and values (for instance, prices or risks – see Chapter 7). 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

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

Couldry, Nick
 categories
 power
 data!training
 machine
 learning!regularization

its massive complexity still appears to us as a *common* world
 (Couldry 2012, 62) ,
 The orderings of practice, effected through categories, undergo a great deal of intensification via machine learning. While Couldry does not in this context discuss data mining, machine learning or predictive analytics, his analysis of categorisation and its contribution to a massively complex but common social world points directly to the order of practice of practice.

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

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

Massive asymmetries in a common world

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

The ordering practices of machine learning display some rather stunning uniformities. Some expert practitioners complain of this uniformity. Jeff Hammerbacher , previously chief research scientist at Facebook, co-founder of a successful data analytics company called Cloudera, and currently working also on cancer research at Mount Sinai hospital, complained about the spread of machine learning in 2011: ‘the of my generation are thinking about how to make people click ads’ (Vance 2011) . Leaving aside debates about the ranking of ‘best minds’ (see chapter 8), Hammerbacher was referring to the flourishing use of predictive analytics techniques in online platforms such as Twitter, Google and Facebook, and on websites more generally, whether they be websites that sell things or advertising space. The mathematical skills of many PhDs from MIT, Stanford or Cambridge were wrangling data in the interests of micro-targeted advertising. As Hammerbacher observes, they were ‘thinking about how to make people

Association Rule
Mining|seemachine
learner!\textit{atapriori}
algorithm
machine learning!as
ordering practice
Hammerbacher, Jeff
advertising, online

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

click ads,’ and this ‘thinking’ mainly took and does take the form of building predictive models that tailored the ads shown on websites to clusters of individual preferences and desires. In thinking about individual people, and indeed, in seeking to figure an individual amidst very large numbers of people (often numbering millions and sometimes hundreds of millions), the ‘best minds’ were also constructing and operating with the very forms of abstraction we see in Equation ??.

Hammerbacher’s unhappiness with ad click prediction resonates in critical responses to machine learning as used in other settings. Some versions of the digital humanities make extensive use of machine learning. To cite one example, *Macroanalysis: Digital Methods and Literary History*, Matthew Jockers describes he or we might relate to one currently popular machine learning or statistical modelling technique, the topic model (itself the topic of discussion in Chapter 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 topic models or Latent Dirichlet Allocation models provide a litmus test of how relations to machine learning is taking shape in the digital humanities. On the one hand, these models promise to make sense of large corpus of documents in terms of underlying themes or ‘topics.’ Latent topics and latent variables are of much more general interest in predictive modelling and classification (as we will, large document collections have long attracted the interest of machine learners). On the other hand, Jockers

signals the difficulties of relating to machine learning when he suggests Galloway, Alex that ‘it is fair to skip the mathematics’ for the sake of ‘the beauty of the decision tree result’. While one part of the humanities and critical social research exhorts closer attention to the mathematical, another averts its gaze in face of their complexity.

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

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

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

Galloway, Alex|)

Even if ‘cultural workers’ do manage to learn to machine learn, and become adept at re-purposing the techniques in the interests of something other than selling things or generating credit scores, what is to be gained by doing so? Galloway suggests that they might actually reinforce power asymmetries and exacerbate the ethical and political challenges posed by machine learning:

But beyond the challenge of unequal talent and resources is the question of critical efficacy. Is it appropriate to deploy positivistic techniques against those self-same positivistic techniques?

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

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

What cannot be automated?

phronesis!predictive

Like Galloway, I'm wary of certain deployments of machine learning, particularly the platform-based deployments and their associated sociality (Gillespie 2010; Van Dijck 2012). In certain cases, they do seem to be 'laying claim to the apparatus of knowledge production.' Yet even amidst the trashy ephemerality of targeted online advertising or the more elevated analytics of literary history, the transformations in knowledge and knowing do not simply align intellectual work and capitalist work. Machine learning as a data practice is not simply automating existing economic relations, even if that reproduction heavily steers its normal practice. While Hamerbacher and Galloway are understandably somewhat dismissive of the existential gratifications and critical efficacy of building targeted advertising systems or document classifiers, the 'deployment' of machine learning is not a finished process, but very much in train, constantly subject to revision, re-configuration and alteration. Could machine learners become engines of difference? Where in the algorithms, calculations, abstractions and regularizing practices of machine learning would differences be re-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 couples critical and practical responses in a situationally aware calculative knowledge-practice, a form of predictive phronesis (Aristotle 1981). For instance, the campaign to re-elect Barack Obama as U.S. President in 2011-12 relied heavily on micro-targetting of voters in the leadup to the election polls (Issenberg 2012; Mackenzie et al. 2016). In response to the data analytics-driven election campaign run by the US Democrats in

natural language
processing
Bogost, Ian
Netflix
Google!Google Flu

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

While these cases may be exceptional achievements, and indeed highlight the suffocating weight of the ad-tech application of machine learning, the proliferation of scientific usage suggests that the generalization of machine

learning cannot be reduced to personalized advertising. Despite their many operational deployments, the coming together of algorithm, calculation and technique in a form of data practice is not fully coherent or complete. In order to qualify or specify how machine learners exist in their generality, we would need to specify their operations at a level of abstraction that neither attributes a mathematical essence to them nor frames them as producers of relative surplus value. Finding ways of accommodating their diversity, loose couplings and mutability would mean grasping their operational power and their capacity to create new forms of difference.⁴

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

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

The term ‘generative’ is somewhat resonant in the field of machine learning as generative models, models that treat modelling as a problem of specifying the operations or dynamics that could have given rise to the observed data, are extremely important. If we consider only Andrew Ng’s CS229 machine learning lectures on Youtube ([Lecture 1 / Machine Learning \(Stanford\) 2008](#)) (lectures I discuss in chapter 2 and draw on throughout this book), we can see that they introduce generative models in Lecture 5 and 6. Although this seems to be only a small part of the 18 lectures given in the course, later lectures on the expectation maximisation algorithm (12-13), and then on unsupervised learning techniques such as factor analysis and principal component analysis, independent component analysis, are also effectively exploring generative models. A similar distribution of topics can be found in *Elements of Statistical Machine Learning*(Hastie, Tibshirani, and Friedman 2009). Generative models, while perhaps slightly less common in practice than discriminative models, nevertheless capture the sense that algorithms are not just implementations of rules for filtering, sorting, or deciding, but carry within them ontological commitments that might actually challenge social theory in interesting ways. In contrast to Lash, I would suggest that the generativity of these algorithms needs to be differentiated from the algorithmic processes that implement rules more generally. Moving into the data via a generative probabilistic model is very different to moving into the data through say a database query. The models, whether generative or discriminative (models such as decision tree, logistic regression or even neural networks that are more limited in their probabilistic underpinnings), are more like meta-algorithms that reorganize other algorithmic processes on varying scales.

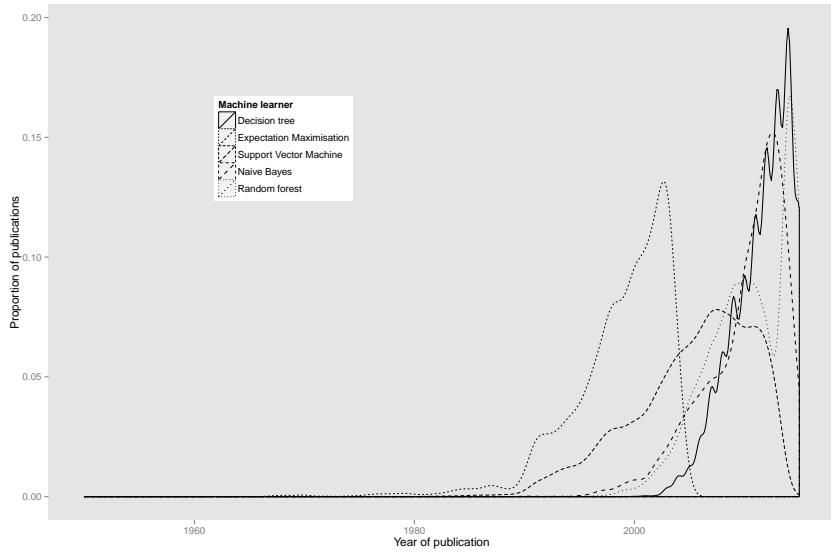


Figure 1.3: Machine learners in scientific literature. The lines in the graph suggest something of the changing fortunes of machine learners over time. The publication data comes from Thomson Reuter's *Web of Science*. Separate searches were run for each machine learner. In these plots, as in the GoogleTrends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

Different abstractions in machine learning?

Table 1.1 presents a small sample of scientific literature at the intersection of ‘climate’ and machine learning. This sample, while no doubt dwarfed by the flood of publications on recommendation systems, targeted advertising or handwriting recognition, is typical of the epistemic atmosphere associated with machine learners. (I return to this topic in Chapter 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.) But the longevity and plurality of experiments, variants, alternative techniques, implementations and understandings associated with machine learning makes it difficult to immediately reduce them to capitalist captures of knowledge production.

Figure 1.3 derives from counts of scientific publications that mention partic-

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

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

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

abstraction!operational practice of abstraction!lived

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

The diagram

Much of this book attempts to identify good levels of abstraction for the data practices of machine learning. The diagram – a form of drawing that smooths away many of the frictions and variations in drawing – anchors much of my account (see chapter 2 for a fuller framing). Diagrams practically abstract. They retain a connection to doing things, such as learning, that other accounts of abstraction sometimes struggle with. Perceptually and technically, they span and indeed criss-cross between human and machine learners. They exhibit compositional characteristics of substitution, variation and superimposition, as well as a play or movement amongst their elements. By virtue of its diagrammatic composition, machine learning might bring something new into the worlds it traverses. If machine learners reorganise data, calculation, classification, decision, control and prediction, that might have some precedent. But precedent or novelty would be quite

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

Above all, diagrams can be implemented in multiple ways. Using implementations, graphical and mathematical forms, books and the heavy accumulation of scientific publications from many disciplines (for instance, as seen in figure 1.3), the book follows six main operations diagrammatically: vectorisation, optimisation, probabilisation, pattern recognition, regularization and propagation. These generic operations compose a diagram of machine learning spanning hardware and software architectures, organizations of data and datasets, practices of designing and testing models, intersections between scientific and engineering disciplines, ongoing historical transformations of notions of pattern, class, rank and order, professional and popular pedagogies, and their associated subject positions. With varying degrees of formalization and consistency, these operations seek to offer an alternative account of machine learning, an account in which some feeling of agency and of affective movement can take route. Where do the operations come from? They as technical processes and practices, sometimes at a quite low

level (for instance, vectorisation) and other times widely distributed (for instance, in pedagogy). They become figures when drawn on a diagram. As an abstraction, the diagram of the operational power of machine learning does not map the footprint of a strategic monolith, but highlights the local relations of force that feed into the generalization and plurality of the field in both its monumental and peripheral variations. These local orderings, distributions, rankings and estimation can support hegemonic platforms, but they may also concatenate in different vectors of movement.

Title	Year
New approach to calculation of atmospheric model physics:	2005
Accurate and fast neural network emulation of longwave radiation in a climate model	
Exit polling in a cold climate: the BBC-ITV experience in Britain in 2005	2008
Independent Component Analysis of Climate Data: A New Look at EOF Rotation	2009
A Streamflow Forecasting Framework using Multiple Climate and Hydrological Models(1)	2009
Strategies for Reforestation under Uncertain Future Climates: Guidelines for Alberta, Canada	2011
The potential use of a Gadget model to predict stock responses to climate change in combination with Bayesian networks: the case of Bay of Biscay anchovy	2011
A weather-type statistical downscaling framework for ocean wave climate	2014
Predicting thermal vulnerability of stream and river ecosystems to climate change	2014
Statistical Postprocessing of High-Resolution Regional Climate Model Output	2015
A bitter cup: climate change profile of global production of Arabica and Robusta coffee	2015

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

Domingos, Pedro
Deleuze, Gilles
diagram

Chapter 2

Diagramming machine learning

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

Diagram: ‘a functioning, abstracted from any obstacle ... or friction’ (Deleuze 1988, 34)

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

programmability!problem
of
abstraction|(
Python|sealsoprogram-
ming
language
R|sealsoprogramming
language|R

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

The changes in data practice associated with machine learning work on multiple levels, ranging from infrastructures and chip architectures through to mathematical functions *and* categorisations with significant social and economic implications. Abstracting as a practice takes many forms and is distributed widely, making it difficult to identify a single level of abstraction. In this chapter, therefore, I attend to the methodological problem of identifying and mapping the different abstractive practices intersecting in machine learning. These practices can be traced through code written to do machine learning, through pedagogical expositions of machine learning focused on both mathematical operations and graphical diagrams, and the forests of scientific or technical research publications, ranging from textbooks to research articles, that vary, explore, experiment and generalize machine learners. Both the coding practices and the pedagogical expositions have shifted substantially over the last decade or so due to the growth in open source programming languages and as a part of the broader and well-known expansion of digital media cultures (Couldry 2012). The fact that scientists, software developers and data analysts use programming languages such as Python and R just as much as commercial statistical and data software packages such as Matlab, SAS or SPSS is perhaps symptomatic of shifts in calculative culture (Muenchen 2014), but almost definitely crucial to the

generalization of machine learning. Scientific research, which has long relied on counting and calculation, increasingly organises and processes data more comprehensively because workflows, datasets, algorithms and databases can be quickly and flexibly formatted and manipulated in code. (Scientific computing languages such as FORTRAN – ‘Formula Translator’ – have long underpinned scientific research and engineering applications in various fields (Campbell-Kelly 2003, 34-35).) Several decades of concentrated research and intensive development of statistical machine learners in science, government, industry and commerce have gradually developed fairly regular operations, patterns and functions for reshaping data and constructing models that classify and predict events and associations between things, people, processes, etc. When Domingos speaks of ‘growing’ programs, he might well be referring to the accumulation of certain well-worn data practices that laminate layers of abstraction. In some ways, the different modes of writing code are precisely the faceted levels of abstraction which we might need to access and traverse in order to know and come to grips empirically with contemporary compositions of power and subjectivity.

programming
language!FORTRAN
abstraction|)
Ng, Andrew

A diagram of the elements of machine learning

In the course of writing this book, as well as reading the academic textbooks, popular how-to books, software manuals, help documents and blog-how-to posts, I attended graduate courses in Bayesian statistics, genomic data analysis, data mining, and missing data. Significantly, I also participated in the online machine learning courses and some machine learning competitions.^[^1.4] These are all typical activities for people learning to do machine learning. Importantly, these courses, books, competitions and programming languages are widely viewed and used. Andrew Ng’s machine learning course on Coursera (of which he is co-founder, along with Daphne

Koller, Daphne
 \textit{Elements of Statistical Learning}
 Hastie, Jeff
 Tibshirani, Rob
 Friedman, Jerome

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

In terms of scientific publications, *Elements of Statistical Learning* is a quasar. It is a massive textual object, densely radiant with diagrams, equations, tables, algorithms, graphs and references to other scientific literature. From the first pages proper of the book, almost every page has a figure or a table or a formal algorithm (counting these together: equations = 968, figures = 291; tables = 34; and algorithms = 94, giving a total of 1387 operational devices threaded through the book). Around 968 equations rivet the text into mathematical abstractions of varying sophistication, and construct an semiotic machinery of considerable sophistication and connectivity. Importantly, on each page of the book we are seeing, reading,

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

puzzling over and perhaps learning from the products of code execution. hyperobject
The figures are all produced by code. The tables are mostly produced by Ripley, Brian
code. The algorithms specify how to implement code, and the equations Mitchell, Tom
diagram various operations, spaces and movements that meant to run as Flach, Peter
code.

In the range of references, it's combinations of code, diagram, equation, scientific disciplines and computational elements, and perhaps in the somewhat viscous, inter-objectively diverse referentiality that impinges on any reading of it, machine learning betrays some hyperobject-like tendencies (Morton 2013). Like the online machine learning courses and programming books I discuss below, *Elements of Statistical Learning* combines statistical science with various algorithms to 'learn from data' (Hastie, Tibshirani, and Friedman 2009, 1). 768 tersely written and quite mathematical pages range across various kinds of problems (identifying spam email, predicting risk of heart disease, recognising handwritten digits, etc.), and various machine learning techniques, methods and algorithms (linear regression, k-nearest neighbours, neural networks, support vector machines, the Google Page Rank algorithm, etc.). This is not the only juggernaut machine learning text. I could just have well cleaved to Alpaydin's *Introduction to Machine Learning* (Alpaydin 2010) (a more computer science-base account), Christopher Bishop's heavily mathematical *Pattern recognition and machine learning* (Bishop 2006), Brian Ripley's luminously illustrated and almost coffee-table formatted *Pattern Recognition and Neural Networks* (Ripley 1996) , Tom Mitchell's earlier artificial intelligence-centred *Machine learning* (Mitchell 1997) , Peter Flach's perspicuous *Machine Learning: The Art and Science of Algorithms that Make Sense of Data* (Flach 2012) , or further afield, the sobering and laconic *Statistical Learning for Biomedical Data* (Malley, Malley, and Pajevic 2011). These and quite a few other

recent machine learning textbooks display a range of emphases, ranging from the highly theoretical to the very practical, from an orientation to statistical inference to an emphasis on computational processes, from science to commercial applications.

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

Who reads the book? It is often cited by academic machine learning practitioners as an authoritative guide. On the other hand, students participating in new data science courses often come from different disciplinary backgrounds and find the tome unhelpful (see the comment by students during an introductory data science course documented in (Schutt and O’Neil 2013)). Whether the citations are friendly or not, Google Scholar reports over 20,00 citations of the book <http://scholar.google.com/scholar?hl=en&q=elements+of+statistical+machine+learning>, October 2014), a huge citation count by any standards. (Michel Foucault’s *The History of Sexuality: an Introduction*, one of the most highly cited book in the humanities, receives

about the same number of citations.) The citations that (Hastie, Tibshirani, and Friedman 2009) as well as its previous edition (Hastie, Tibshirani, and Friedman 2001) receive do not arrive principally from machine learning researchers. They come from a wide variety of fields. It is hard to find a field of contemporary science, engineering, natural, applied, health and indeed social science that has not cited it. A Thomson-Reuters Scientific ‘Web of Science’(TM) search for references citing either the first or second edition of (Hastie, Tibshirani, and Friedman 2009) yields around 9000 results. These publications sprawl across well over 100 different fields of research. While computer science, mathematics and statistics dominate, a very diverse set of references comes from disciplines from archaeology, through fisheries and forestry, genetics, robotics, telecommunications and toxicology ripple out from this book since 2001. Table 2.1 shows the top 20 fields by count. One could learn something about the diagrammatic movement of machine learners from that reference list, which itself spans biomedical, engineering, telecommunications, ecology, operations research and many other fields. While it is not surprising to see computer science, mathematics and engineering appearing at highest concentration in the literature, the molecular biology, control and automation, operation research, business and public health soon appear, suggesting something of the propagating energy of machine learning.

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

Non-negative matrix
factorization
'kittydar'
face recognition

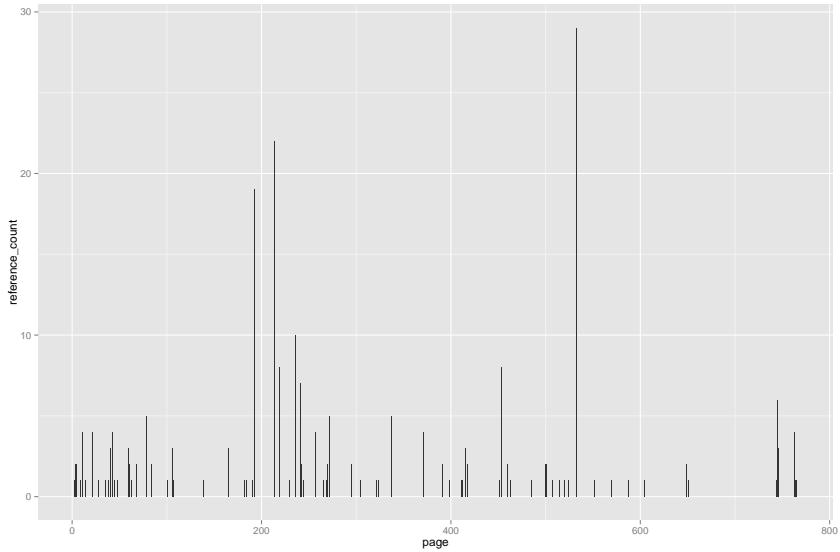


Figure 2.1: Pages cited from *Elements of Statistical Learning* by academic publications in all fields.

distinct pages are referenced in the citing literature. As figure ?? indicates, certain portions of the book are much more heavily cited than others. This distribution of page references in the literature that cites *Elements of Statistical Learning* is a rough guide to how the book has been read in different settings. For instance, the most commonly cited page in the book is page 533. That page begins a section called ‘Non-negative Matrix Factorization’, a technique frequently used to process digital images to compress their visual complexity into a simpler set of visual signals (Hastie, Tibshirani, and Friedman 2009, 553). (The underlying reference here is the highly cited (Lee and Seung 1999)) It is particularly powerful because it, as Hastie and co-authors write, ‘learns to represent faces with a set of basis images resembling parts of faces’ (Hastie, Tibshirani, and Friedman 2009, 555). (So **kittydar**, which doesn’t use NMF, might do better if it did, because it could work just with parts of the images that lie somewhere near the parts of a cat’s face – its nose, its eyes, its ears.)

Conversely, what do the authors of *Elements of Statistical Learning* read?

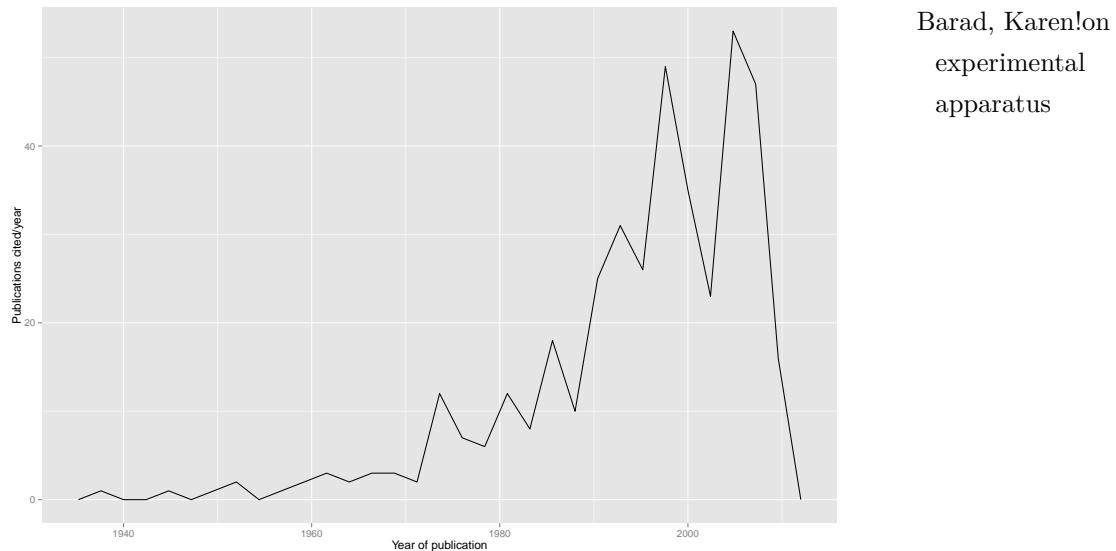


Figure 2.2: Publications cited by *Elements of Statistical Learning*. The references range over almost 80 years, with peaks in late 1970s, late 1980s, mid-1990s and mid-2000s. These peaks relate to different mixtures of cybernetics, statistics, computer science, medicine, biology and other fields running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

The semiotic machinery of the book relies on vectors and orbital trajectories that converge from many different directions. The hyperobject-like aspect of the book comes from the thick weave of equations, diagrams, tables, algorithms, bibliographic apparatus, and numbers wreathed in typographic ornaments drawn from many other sources. For instance, in terms of outgoing references or the literature that it cites, *Elements of Statistical Learning* webs together a field of scientific and technical work with data and predictive models ranging across half a century. The reference list beginning at page 699 (699) runs for around 35 pages, and the five hundred or so references there point in many directions. The weave of these elements differs greatly from documents in the humanities or social sciences, and, almost before anything else, prompts attention to the problem of reading parts and fragments, and relating to an object or perhaps an apparatus in the sense that Karen Barad ascribes to experimental setups (Barad 2007).

Kuhn, Thomas
phronesis

As Figure 2.2 indicates, the citational fabric of *Elements of Statistical Learning* is woven with different threads, some reaching back into early twentieth statistics, some from post-WW2 cybernetics, many from information theory and then in the 1980s onwards, increasingly, from cognitive science and computer science. While many of these references either point to Hastie, Tibshirani or Friedman’s own publications, or that of their statistical colleagues, the references show that these authors are also roving quite widely in other fields and over time. *Elements of Statistical Learning* as a text is processing, assimilating and recombining techniques, diagrams and data from many different places and times. Both the inward and outward movements of citation suggest that the book, like much in field of machine learning, has a wave function or matrix-like character that constantly superimposes and transposes elements across boundaries and barriers. The implication here is that machine learning as a field has a highly interwoven texture, and in this respect differs somewhat from the classical understandings of scientific disciplines as bounded by communities of practice, norms and problems (as for instance, in Thomas Kuhn’s account of normal science (Kuhn 1996)). This aggregate or superimposed character of machine learning should definitely figure in any sense we make of it, and will inevitably affect how we phronetically relate to it. . Hence, the different waves appearing in the references cited in (Hastie, Tibshirani, and Friedman 2009) will shape discussion in later chapters in certain ways (for instance, biology is the topic of chapter 7 and optimization functions of chapter 4).

The obdurate mathematical glint of machine learning

linear regression model | (

linear regression model

While references from many different places go in and out of *Elements of Statistical Learning*, they are nearly all articulated in mathematical form. The mechanics of machine learning as a level of abstraction are mathematical, and these mechanics soon begin to operate in the vast diagram the book presents. Given that mathematics is itself diverse and multi-stranded, what kind of mathematics matters most here? The predictive models in *Elements of Statistical Learning* are dominated by a single prediction technique: linear regression models or fitting a line to points. The linear regression model is pivotal, not just in *Elements of Statistical Learning* but in much of the scientific and engineering literature. (We have already seen something of this model in the Introduction, in the equations for linear regression ??). The linear regression model pushes up some of the citational peaks in Figure 2.2. Even though it is an old technique dating back to Francis Galton in the 1890s (see (Stigler 1986, chapter 8)), drawing sharp, straight lines through opaque clouds of data-matter is what much machine learning still seeks to do. It manoeuvres in complex ways in drawing these lines (as we will see), but lines cutting, cleaving, and shattering things are very much the central working abstraction of machine learning. Hastie, Tibshirani and Friedman acknowledge the statistical legacy and inheritance in machine learning:

The linear model has been a mainstay of statistics for the past 30 years and remains one of our most important tools. Given a vector of inputs $X^T = (X_1, X_2, \dots, X_p)$, we predict the output Y via the model

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

The term $\hat{\beta}_0$ is the intercept, also known as the *bias* in machine

linear regression model|) learning (Hastie, Tibshirani, and Friedman 2009, 11).

In the course of the book, linear regression is subjected to countless variations, iterations, expansions and modifications. Their own research spans several decades and a range of topics concerning linear regression models and their variations ('ridge regression'; 'least angle regression'; etc.). But this introduction of the 'mainstay of statistics,' the linear model, already introduces a harsh form – the mathematical equation – that is perhaps the most prominent feature in the text. Any reading of the book has to work out a way to traverse the forms show in equation 8.2.

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

In its relatively compressed typographic weave, expressions such as Equation 8.2 operationalize movements through data that call for some attention. These expressions, which are not comfortable reading by and large for non-technical readers, are however worth looking at carefully if we want to move 'at the same level of abstraction as the algorithm' (Pasquinelli 2015). They can be found in hundreds in (Hastie, Tibshirani, and Friedman 2009), but also in many other places. While their presence distinguishes machine learning from many much computer science where mathematical equations are less common, these equations also allow the book to collate and borrow from a panoply of scientific publications and datasets in fields of statistics, artificial intelligence and computer science. Along with the citations, the graphical plots, the algorithms (implemented in code described below), these equations are integral connective tissue in machine learning. Unless we come to grips with their diagram force, without succumbing to the obscuring dazzle of mathematical abstraction, the connectivity and mobility of these forms will be lost on us.

I find it useful here to follow Charles Sanders Peirce account of mathematics Peirce, C.S. in terms of diagrams. ‘Mathematical reasoning,’ he writes, ‘is diagrammatic’ diagram|(Charles S. Peirce 1998, 206). That it, we should see mathematics, whether diagram!icon it takes an algebraic or geometrical form, whether it appears in symbols, diagram!indexical sign letters, lines or curves as diagrams. Now for Peirce, a diagram is a kind of ‘icon.’ The icon is a sign that resembles the object it refers to: it has a relation of likeness. What likeness appears in 8.2? As Peirce says, ‘many diagrams resemble their objects not all in their looks; it is only in respect to the relations of their parts that their likeness consists’ (Charles S. Peirce 1998, 13). As we will soon see, 8.2 could be expressed in statements in a programming language like Python or R, or in algorithmic pseudo-code, or perhaps most accessibly, as graphic figure (a line drawn through a cloud of points). In none of these associated diagrams can the relations between the parts be observed in the same way as they in the algebraic form. The ‘very idea of the art’ as Peirce puts it (Charles Sanders Peirce 1992, 228) of algebraic expressions is that the formulae can be manipulated. The graphic form of the expression include the various classical Greek symbols such as Σ or Π , as well as the letters x, y, z and the indices (indexical signs) that appear in subscript or superscript, as well as the spatial arrangement of all these in lines and sometimes arrays. A variety of relations run between these different symbols and spatial arrangements. For instance, in all such expressions, the difference between the left hand side of the ‘=’ and the right hand side is very important. By convention, the left hand side of the expression is the value that is predicted or calculated (the ‘response’ variable) and the right hand side are the input variables or ‘features’ that contribute data to the model or algorithm. This spatial arrangement fundamentally affects the design of algorithms. In the case of 8.2, the ‘ \hat{Y} ’ over \hat{Y} symbolises a predicted value rather than a value that can be known completely through deduction, derivation or calculation. This distinction between predicted

diagram|) and actual values organizes a panoply of different practices and imperatives (for instance, to investigate the disparities between the predicted and actual values – machine learning practitioners spend a lot of time on that problem).

The general point is that the whole formulae is a diagram, or an icon that ‘*exhibits*, by means of the algebraical signs (which are not themselves icons), the relations of the quantities concerned’ (Charles S. Peirce 1998, 13). Because such diagrams suppress so many details, they allow one to focus on a more limited range of relations between parts. The manipulation of those relations generates new diagrams or patterns. This affordance of diagrammatic forms is extremely important in the intensification of machine learning. Importantly, diagrams can diagram other diagrams. Put differently, operations can be themselves the subject of operations. Or functions can themselves for functions of functions. This nesting and coiled aspect of the diagrams is highly generative since it allows what Peirce calls ‘transformations’ (Charles S. Peirce 1998, 212) or the construction of ‘a new general predicate’ (Charles Sanders Peirce 1992, 303).² The intensive processing of data today via predictive models is largely channelled via such diagrams. These diagrams are not conspicuous in the infrastructures, and they are not directly seen by people or things they impinge upon even as they have their effect.

2. Felix Guattari makes direct use of Peirce’s account of diagrams as icons of relation in his account of ‘abstract machines’ (Guattari 1984). He writes that ‘diagrammaticism brings into play more or less de-territorialized trans-semiotic forces, systems of signs, of code, of catalysts and so on, that make it possible in various specific ways to cut across stratifications of every kind’ (145). Here the ‘trans-semiotic forces’ include mathematical formulae and operations (such as the banking system of Renaissance Venice, Pisa and Genoa). They are trans-semiotic because they are not tethered by the signifying processes that code experience or speaking positions according to given stratifications such as class, gender, nation and so forth. While Guattari (and Deleuze in turn in their co-written works (Guattari and Deleuze 1988)) is strongly critical of the way which signification territorializes (we might think of cats patrolling, marking and displaying in order to maintain their territories), he is much more affirmative of diagrammatic processes. He calls them ‘a-signifying’ to highlight their difference from the signifying processes that order social strata. He suggests that diagrams become the foundation for ‘abstract machines’ and the ‘simulation of physical machinic processes.’ Writing in the 1960s, Guattari powerfully anticipates the abstract machines and their associated diagrams that have taken shape and physical form in the succeeding decades.

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

CS229, 2007: returning again and again to certain features

If we were to follow Peirce’s injunction to ‘return again and again to certain features,’ how would we do that? *Elements of Statistical Learning* is a difficult book to read in isolation (although it does repay re-reading). The cost of the diagrammatic density of its ‘elements’ (equations, citations, tables, datasets, plots) is a certain refractory feeling of ‘not quite understanding’ for many readers. This feeling is inevitable because the book condenses finished work from several disciplines, and partly because it seeks to frame a great diversity of materials *statistically*. (The statistical aspects of machine learning are the main topic of chapter 5). Professor Andrew Ng’s course

Ng, Andrew
linear regression

‘Machine Learning’ CS229 at Stanford (<http://cs229.stanford.edu/>) might provide a supplementary path into machine learning (*Lecture 1 / Machine Learning (Stanford) 2008*).³ Note that Ng is a computer scientist, not a statistician. The course description runs as follows:

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

CS229 is in many ways a typical computer science pedagogical exposition of machine learning. Machine learning expositions usually begin with simple datasets and the simplest possible statistical models and machine learning algorithms (usually linear regression), and then, with a greater or lesser degree of attention to issues of implementation, move through a succession of increasingly sophisticated and specialised techniques. This pattern is found in many of the how-to books, in the online courses, and in the academic textbooks, including (Hastie, Tibshirani, and Friedman 2009). Ng’s CS229 lectures differ from almost all other expository materials in that we see someone writing and drawing line after line of equations using chalk on a blackboard. Occasionally, questions come from students in the audience (not shown on the Youtube videos), but mostly Ng’s transcription of equations and other diagrams from the paper notes he holds to blackboard continues

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

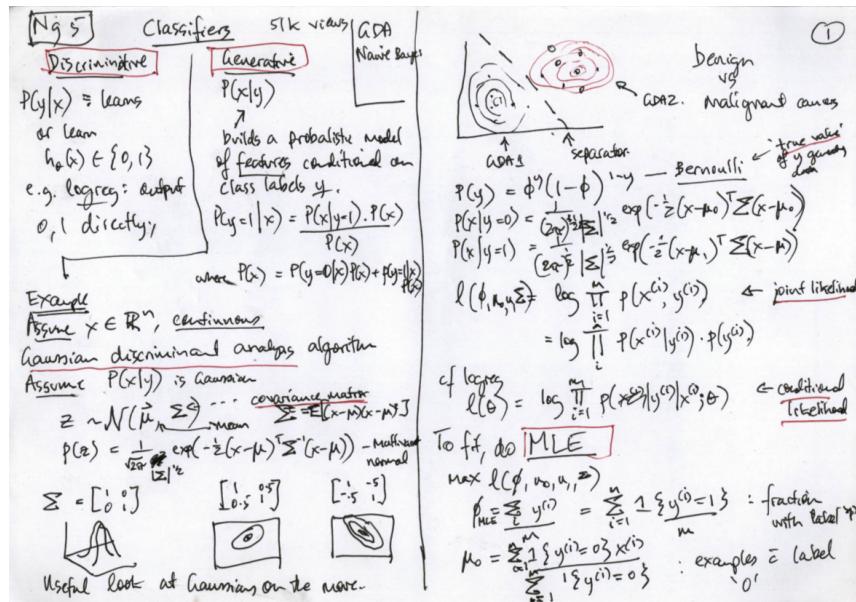


Figure 2.3: Class notes lecture 5, Stanford CS229, 2007

uninterrupted.⁴

Ng's YouTube anachronistic but popular lectures have a certain diagrammatic density that comes from the many hours of chalked writing he stages during the course. In a time when PowerPoint presentations or some other electronic textuality would very much have been the norm (2007), why is a Stanford computer science professor, teaching a fairly advanced postgraduate course, writing on a chalkboard by hand? (Figure 2.3 shows a brief portion of around 100 pages of notes I made on this course.) The act of writing down these equations and copying the many hand-drawn graphs Ng produced was a deliberative descriptive experiment, but more importantly an exercise in 'returning again and again' to what is perhaps overly hardened

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

diagram!hand-drawn

in *Elements of Statistical Learning*. Like the 50,000 or so other people who had watched this video, I complied with Ng's injunction to 'copy it, write it out, cover it, and see if you can reproduce it' ([NgAndrew_2008](#)). While it occasions much writing and drawing, and many struggles to keep up with the diagrams that Ng narrates as he writes, it seems to me that this writing of equations, with all their substitutions and variations, alongside the graphic sketches of intuitions about the machine learning techniques, adds something of the *derivations* that is quite hard to finesse in *Elements of Statistical Learning*. There the diagrammatic weave between the expressions of linear algebra, calculus, statistics, and the algorithms is almost too tight to work with. In Ng's CS229 lectures, by contrast, the weaving, while still complex, is much more open. They lack the citational tapestry of (Hastie, Tibshirani, and Friedman [2009](#)), they are not able to wield the datasets and the panoply of graphic forms found there, and virtually no machine learning code appears on the blackboard (although the CS229 student assignments, also to be found online, are code implementations of the algorithms and models), but Ng's lectures have a useful thawing effect. As we will see, only by tracking some of the movements of code that underpin machine learning do we identify levels of abstraction that we might want to think about.

Some broadly shared topic structures help in any navigation of these pedagogical literatures. The textbooks, the how-to recipe books ([Segaran 2007](#); [Kirk 2014](#); [Russell 2011](#); [Conway and White 2012](#)) and the online university courses on machine learning often have a similar topic structure. They nearly always begin with 'linear models' (fitting a line to the data), then move to logistic regression (a way of using the linear model to classify binary outcomes; for example, spam/not-spam; malignant/benign; cat/non-cat), and afterwards move to some selection of neural networks, decision trees,

support vector machine and clustering algorithms. They add in some decision theory, techniques of optimization, and ways of selecting predictive models (especially the bias-variance tradeoff).⁵ The topic structures have in recent years started to become increasingly uniform. This coagulation around certain topics, problems and mathematical formalisms is both something worth analyzing (since, for instance, it definitely affects how machine learning is taken up in different settings), but should not be taken as obvious or given. In this respect, Ng's step-by-step handcrafted derivations are too derivative to really track the trans-semiotic intensities of equations such

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

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

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

learning as equation 8.2. How would one learn to read such diagrams less linearly (Ng's long columns of algebraic derivation), more diagonally and in better alignment with the multi-faceted abstraction they often compact?

The learning of machine learning

For a book with ‘learning’ in its title, *Elements of Statistical Learning* has only a little to say explicitly about how to learn machine learning. ‘Learning’ is briefly discussed on the first page of *Elements of Statistical Learning*, but the book hardly ever returns to the topic or even that term explicitly. We learn on page 2 that a ‘learner’ in machine learning is a model that predicts outcomes. (As I discuss in chapter 4, learning is comprehensively understood in machine learning as finding a mathematical *function* that could have generated the data, and optimising the search for that function as much as possible.) The notion of learning in machine learning derives from the field of artificial intelligence. The broad project of artificial intelligence, at least as envisaged in its 1960s-1970s heyday as a form of symbolic reasoning, is today largely regarded as a deadend. There is no general, or comprehensive artificial intelligence in existence, even though many efforts continue to develop ‘deep learners’ (see for instance, Google Corporation’s ongoing research into ‘deep learning’ of its own data; and chapter 8). If a general intelligence did not appear, certainly a lot of learning was undertaken. The field of machine learning might be seen as one such offshoot since it paused the process of machines becoming intelligences at a developmental phase that entails much close supervision, monitoring and oversight.

The so-called ‘learning problem’ and the theory of learning machines developed largely by researchers in the 1960-1970s was largely based on work already done in the 1950s on cybernetic devices such as the perceptron,

the prototypical neural network model developed by the psychologist Frank Rosenblatt in the 1950s (Rosenblatt 1958). Drawing on the McCulloch-Pitts model of the neurone, Rosenblatt implemented the perceptron, which today would be called a single-layer neural network (Hastie, Tibshirani, and Friedman 2009, 393) on a computer at the Cornell University Aeronautical Laboratory in 1957. For present purposes, it is interesting to see what diagrams Rosenblatt used and how they differ from contemporary diagrams.⁶

What has been *learned* in the intervening period?

If we compare the diagram of Rosenblatt's perceptron shown in Figure 2.4a to the typical contemporary diagram of a neural network shown in Figure 2.4b, the differences are not that great in many ways. The diagram of a neural network found in Rosenblatt's paper (Rosenblatt 1958) has no mathematical symbols on it, but the one from (Hastie, Tibshirani, and Friedman 2009) does. Rosenblatt's retains neuro-cognitive-anatomical reference points (retina, association area, projection area) whereas Hastie

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

Sometimes machines are the very means by which we can stay alive psychically, and they can just as readily be a means for affective expansion and amplification as for affective attenuation. This is especially the case of computational machines (Wilson 2010, 30).

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

Vapnik, Vladimir
machine
learner!perceptron|)

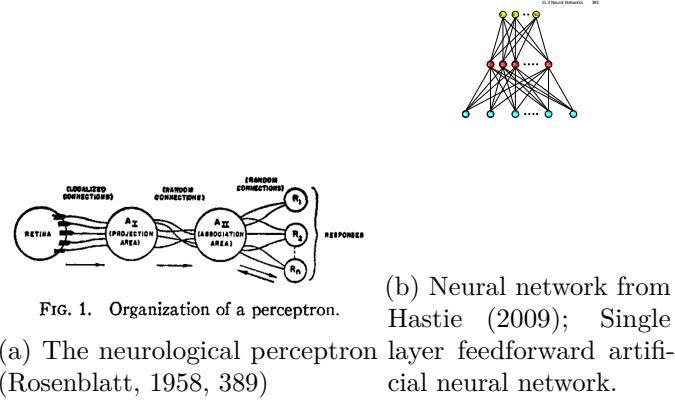


FIG. 1. Organization of a perceptron.

(a) The neurological perceptron layer feedforward artifi-

(b) Neural network from
Hastie (2009); Single
layer feedforward artifi-

(Rosenblatt, 1958, 389)

cial neural network.

Figure 2.4: 1958 perceptron and 2001 neural net compared

et. al.'s replaces them with the symbols that we have already seen in play in the expression for the linear model 8.2. What has happened between the two diagrams? As Vladimir Vapnik, a leading machine learning theorist, observes: 'the perceptron was constructed to solve pattern recognition problems; in the simplest case this is the problem of constructing a rule for separating data of two different categories using given examples' (Vapnik 1999, 2). While computer scientists in artificial intelligence of the time, such as Marvin Minsky and Seymour Papert, were sceptical about the capacity of the perceptron model to distinguish or 'learn' different patterns (Minsky and Papert 1969), later work showed that perceptrons could 'learn universally'⁷.

7. In describing the entwined elements of machine learning techniques, and citing various machine learning textbooks, I'm not attempting to provide any detailed history of their development. My account of these developments does not explore the archives of institutions, laboratories or companies where these techniques took shape. It derives much more either from following citations back out of the highly distilled textbooks into the teeming collective labour of research on machine learning as published in hundreds of thousands of articles in science and engineering journals, or from looking at, experimenting with and implementing techniques in code. For instance, (Olazaran 1996) offers a history of the perceptron controversy from a science studies perspective. During the 1980s, artificial intelligence and associated approaches (expert systems, automated decision support, neural networks, etc.) were a matter of some debate in the sociology and anthropology of science. The work of Harry Collins would be one example of this (Collins 1990), Paul Edwards on artificial intelligence and Cold War (Edwards 1996), Nathan Ensmenger on chess (Ensmenger 2012), and Lucy Suchman on plans and expert systems (L. A. Suchman

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

What the perceptron latches onto

I am suggesting, then, that we should follow the transformations associated with machine learning diagrammatically, provided we maintain a rich un-

1987) would be others. Philosophers such as Hubert Dreyfus (*What Computers Can't Do* (Dreyfus 1972, 1992) had already extensively criticised AI. In the 1990s, the appearance of new forms of simulation, computational fields such as a-life and new forms of robotics such as Rodney Brook's much more insect-like robots at MIT attracted the interest of social scientists (Helmreich 2000) amongst many others. Sometimes this interest was critical of claims to expertise (Collins), and at other times, interested in how to make sense of the claims without foreclosing their potential novelty (Helmreich). By and large, I don't attend to controversies in machine learning as a scientific field, and I don't directly contest the epistemic authority of the proponents of the techniques. I share with Lucy Suchman an interest in how the ‘effect of machine-as-agent is generated’ and in how ‘translations ... render former objects as emergent subjects’(L. Suchman 2006, 2). I diverge around the site of empirical attention. I'm persevering with the diagrams in each of the following chapters in order to track the movement of tendencies that are not so visible in terms of either the controversies or the assumptions of agency embodied in many AI systems of the 1980s. The agency of machine learning, in short, might not reside so much in any putative predictive or classificatory power it manifests, but rather its capacity to mutate and migrate across contexts.

diagram|(
derstanding of diagram, and remain open to multiple levels of abstraction. Following Peirce, we might begin to see machine learning as a diagrammatic practice in which different semiotic forms – lines, numbers, symbols, operators, patches of colour, words, images, marks such as dots, crosses, ticks and arrowheads – are constantly connected, substituted, embedded or created from existing diagrams. The diagrams we have already seen from *Elements of Statistical Learning* - algebraic formulae and network topology - don't exhaust the variations at all. Just a brief glance through this book or almost any other in the field shows not only many formulae, but tables, matrices, arrays, line graphs, contour plots, scatter plots, dendograms and trees, as well as algorithms expressed as pseudo-code. The connections between these diagrams are not always very tight or close. Learning to machine learning (whether you are a human learner or a learner in the sense of a machine) means dancing between diagrams. This dance is relatively silent and sometimes almost motionless as signs slide between different diagrammatic articulations. Diagrammatization offers then a way to track the ongoing project which tries treat data like farmers treat crops (see epigraph from Domingos in this chapter). To understand what machines can learn, we need to look at how they have been drawn, designed, or formalised. But what in this work of designing and formalising predictive models is like farming? Some very divergent trajectories open up here. On the one hand, the diagrams become machines when they are implemented. On the other hand, the machines generate new diagrams when they function. We need to countenance both forms of movement in order to understand any of the preceding diagrams – the algebraic expressions or the diagrams of models such as the perceptron or its descendants, the neural network. This means going downstream from the textbooks into actual implementations and places where people, algorithms, and machines mingle more than they do in the relatively neat formality of the textbooks. Rather than history

or controversies in the field, I focus on the migratory patterns of methods, diagram[1] and the many configurational shifts associated with their implementations Pearson, Karl as the same things appears in different places. artificial intelligence

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

Table 2.2: The Boolean function NOT-AND truth table

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

artificial intelligence

For present purposes, these kinds of logical and symbolic diagrams have triple relevance. First, the spatial arrangement of the table is fundamental to not only machine learning and most contemporary data practice. (Transformations in tables are the main topic of chapter 3.) That spatial form is diagrammatic in various ways, but principally through the horizontal and vertical relationships it installs. Most datasets come as tables, or end up as tables at some point in their analysis. Second, the elements or cells of this table are numbers. These numbers, 1 and 0 are the binary digits as well as the ‘truth’ values ‘True’ and ‘False’ in classical logic. The truth table for NOT-AND is in some ways typical of data tables because it contain numbers. These numbers are readable as symbolic logical propositions. Hence this table acts as a hinge between numbers and symbolic thought or cognition. (In (Hastie, Tibshirani, and Friedman 2009) most tables contain data in the forms of numbers rather than truth values.) Third, the NAND table in particular has an obvious operational importance in digital logic, since digital circuits of all kinds – memory, processing, and communication – comprise such logical functions knitted together in the intricate fabrics of contemporary calculation.⁸

The obviousness of the logical operation shown in the table stands out. It is hardly surprising to us at all. This looks like the kind of mechanistic calculation that computers do. How, by contrast, could such a truth table be learned by a machine, even a machine whose *modus operandi* and indeed whose very fabric is nothing other than a massive mosaic of such operations inscribed in transistor circuits? Machine learning of such truth tables is not a typical operation today, but does illustrate something of the diagrammatic transformations that classical AI has undergone . If we return to the

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

perceptron, with its neural intuition, how could it learn this kind of logic? It knows nothing of the logical calculus, but it can be induced to take on a logical shape by training it. For instance, a perceptron that ‘learns’ the binary logical operation NAND (Not-AND) is expressed in twenty lines of python code on the Wikipedia ‘Perceptron’ page ([Perceptron 2013](#)). The code is followed by the output – a series of numbers – produced when it runs.

```

threshold = 0.5

learning_rate = 0.1

weights = [0, 0, 0]

training_set = [((1, 0, 0), 1), ((1, 0, 1), 1), ((1, 1, 0), 1), ((1, 1, 1), 0)]

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

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

out = 'weight1, weight2, weight3, error_count, iteration'

print >>outfile, out

iteration_count = 0

while True:

    error_count = 0

    iteration_count += 1

    for input_vector, desired_output in training_set:

        out = ','.join(map(str,weights)) + ',' + str(error_count)

        out = out + ',' + str(iteration_count)

        result = dot_product(input_vector) > threshold

        error = desired_output - result

        print >>outfile, out

```

```

machine
        print >>outfile, out
learner!perceptron|(
        if error != 0:
            error_count += 1
            for index, value in enumerate(input_vector):
                weights[index] += learning_rate * error * value
        if error_count == 0:
            break
outfile.close()

```

What does this code show or say? First of all, we should note the relative conciseness of the code vignette. In citing this code, I'm not resorting to a technical publication or scientific literature as such, or even to a software library or package, just to that popular yet incredibly heavily used epistemic resource, the Wikipedia page, and the relatively generic and widely used programming language `python`.⁹ Whatever the levels of abstraction associated with machine learning, it is hardly ever hermetically opaque. While perceptrons and neural networks are the topic of later chapters, the typical features of this code as the implementation of a machine learning algorithm are the presence of the ‘learning rate’, a ‘`training_set`’, ‘`weights`’, an ‘error count’, and a loop function that multiplies values (‘`dot_product`’). Some of the names such as `learning_rate` or `error_count` present in the code bear the marks of the theory of learning machines that we will discuss. Much of the code here is much more familiar, generic programming. It defines variables, sets up data structures (lists of numerical values), checks conditions, loops through statements or prints results. Executing this code

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

(by copying and pasting it into a python terminal, for instance) produces several dozen lines of numbers that are initially different to each other, but that gradually converge on the same values (see printout). These numbers are the ‘weights’ of the nodes of the perceptron as it iteratively learns to recognise patterns in the input values. None of the workings of the perceptron need concern us at the moment. It is a typical machine learning algorithm, almost always included in machine learning textbooks and usually taught in introductory machine learning classes. Perhaps more strikingly than its persistence as an algorithm over half a century, the implementation of the whole algorithm in twenty lines of code on a Wikipedia page suggests the mobility of these methods. What required the resources of a major university research engineering laboratory in the 1950s can be re-implemented by a cut and paste operation between Wikipedia pages and a terminal window on a laptop in 2014. This is now a familiar observation, and perhaps not very striking at all. Again, what runs across all of these observations are the numbers that the algorithm produces. The NAND truth table has been re-drawn as a list of tuples or sets (see line 4 of the code that defines the variable `training_set`). The perceptron has learned the truth table by being given it as a set of training examples, and then adjusting its internal model – the weights that are printed during each loop of the model as the output – repeatedly until the model is producing the correct values of the truth table. The algorithm exits its main loops (`while True:`) when there are no errors. The effect here is recursive: a model or algorithm implemented in digital logic has learned a basic rule of digital logic at least approximately. This transformation in learning style is symptomatic of the broader transformations that machine learning latches onto. The learning done in machine learning has few cognitive or symbolic underpinnings. It differs from classical AI in that it takes existing symbolic and, increasingly, signifying processes (such as the cat faces that `kittydar`

machine
 learner!perceptron|)
 learning
 Domingos, Pedro

tries find), and latch onto them diagrammatically. The diagrammatic dance between different algebraic, geometrical, algorithmic and tabular forms of expression is where the learning occurs.

At the same time, the perceptron algorithm produces numbers - 0.79999, 2.0 – as weights. These numbers display no direct correspondence with the symbolic categories of boolean True and False or the binary digits 1 and 0. There may be a relation but it is not obvious at first glance. The problem of mapping these calculated numbers – and they truly abound in machine learning – triggers many different diagrammatic movements in *Elements of Statistical Learning* and like-minded texts (and these different movements will be discussed in chapters 5 and 6). These numbers engender much statistical ratiocination (see chapter 5), but here we need only note the distance between symbolically organised diagrams like the NAND truth table of Table 2.2 and the diagrams of a machinic configuration printed as weights in table 2.3.

Machine learning implemented as program

Pedro Domingos refers to change from building programs to growing knowledge (see the epigraph to this chapter). Does this contrast help make sense of the perceptron’s operations or *Elements of Statistical Learning* more generally? The final part of my reading protocol for multi-faceted abstractions in machine learning entails, I am suggesting, tracking the transformations between table 2.2 and table 2.3 in and by program code. What would we learn by studying and implementing machine learners as programs rather than their history or the controversies associated with them? It is not the case that code offers privileged access to abstraction. But it is highly diagrammatic and in sometimes latent ways, affectively participatory. That is, it

expresses the multi-faceted abstraction of machine learning and allows many connections running across facets to be followed. Science studies scholars such as Anne-Marie Mol have urged the need to keep practice together with theories of what exists. Towards the beginning of *The Body Multiple: Ontology in Medical Practice* (Mol 2003), Mol writes:

If it is not removed from the practices that sustain it, reality is multiple. This may be read as a description that beautifully fits the facts. But attending to the multiplicity of reality is also an act. It is something that may be done – or left undone (Mol 2003, 6)

Mol's work offers a cogent case for developing accounts of what is real steeped in the practices that make it real. While similar sounding affirmations of the underpinning role of practice can be found in many parts of social sciences and humanities (since who would not affirm the centrality of practice?), Mol's insistence on this nexus of practice or doing and the existence of things in their plurality offers another way forward in reading *Elements of Statistical Learning*. Tracking techniques – a more stabilised form of practice – and the flow of their implementations is a way of keeping abstractions and calculations in their faceted multiplicity. The media theorist Ian Bogost speculates and practices coding as a way to make sense of the world: 'source code itself often offers inroads in alien phenomenology - particular when carpentered to reveal the internal experience of withdrawn units' (Bogost 2012, 105) . In relation to machine learning, reading and writing code alongside scientific papers, Youtube lectures, machine learning books and competitions is not only a form of observant participation, but directly forms part of the diagrammatic multiplicity. We can see this in a number of different ways.

machine
learners!programs
Mol, Anne-Marie
practice
Bogost, Ian
source code
programming
language|R|()

programming languages First, although barely a single line of code appears in *Elements of Statistical Learning*, nearly all of the examples in *The Elements of Statistical Learning* are implemented in a single programming language, R. The authors say ‘we used the R and S-PLUS programming languages in our courses’ (Hastie, Tibshirani, and Friedman 2009, 9) but the diagrammatic movements of the book owe much more to R code.¹⁰ The proliferation of programming languages such as FORTRAN (dating from the 1950s), C (1970s), C++ (1980s), then Perl (1990s), Java (1990s), Python (1990s) and R (1990s), and computational scripting environments such as Matlab, multiplied the paths along which implementation of machine learning techniques could proceed. It would be difficult to comprehend the propagation of machine learners across domains of science, business and government without paying attention to coding practices. Even if textbooks and research articles are not read, software packages and libraries for machine learning are used. They have a mobility that extends the diagrammatic practices of machine

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

learning into a variety of settings and places where the scientific reading apparatuses of equations, statistical plots, and citations of research articles would not be operative.

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

\begin{table}[!htb]

	Package	How often depended on
methods	methods	74
stats	stats	28
survival	survival	26
MASS	MASS	25
mvtnorm	mvtnorm	21
Matrix	Matrix	12
ggplot2	ggplot2	9
grid	grid	8
igraph	igraph	8
lattice	lattice	8
rJava	rJava	8
rgl	rgl	7
tcltk	tcltk	7
XML	XML	7
utils	utils	6
nlme	nlme	5
ape	ape	4
coda	coda	4
gtools	gtools	4
Rcpp	Rcpp	4

Marx, Karl

```
\caption[\texttt{ElemStatLearn} depends on R packages]{R packages depended on  
by the \texttt{ElemStatLearn} package} \end{table}
```

Second, particular code elements such as R packages relate to many others, and these relations can be used to identify how different facets and levels of abstraction come together. For instance, the line of code shown above

when executed opens another way of reading *Elements of Statistical Learning* and getting a feel for the dragnet of practice running through it. There is nothing too opaque, I would suggest, about the line of code itself, but there are some folds and code-specific convolutions associated with it that point to infrastructural undersides. Take the part of the line `dependencies = 'Suggests'`. When the line of code executes, this stipulation of `dependencies` leads to a quite wide-ranging installation event. If the installation works (and that assumes quite a lot of configuration and installation work has already taken place; for instance, installing a recent version of the R platform), then *Elements of Statistical Learning* is now augmented by various pieces of code, and by various datasets that in some ways echo or mimic the book but in other ways extend it operationally (see tables 2 and ??).¹¹ These code elements are often stunningly specialised. As Karl Marx wrote of the 500 different hammers made in Birmingham, ‘not only is each adapted to one particular process, but several varieties often serve exclusively for the different operations in one and the same process’ (Marx 1986, 375) . Something similar holds in R: thousands of software packages in online repositories suggest that a highly specialised

11. Most of the packages associated with the `ElemStatLearn` implement methods or techniques developed by Hastie, Tibshirani or Friedman, but some are much more generic. MASS for instance is highly cited R package. (Of the 7092 packages in the R CRAN system, 0 depend on the library MASS, itself an adjunct to the influential and highly cited *Modern Applied Statistics with S* (W. N. Venables and Brian D. Ripley 2002), a textbook that presents many machine learning techniques using S, AT & T Bell Labs commercial precursor to the open sourced R.) For our purposes, this hardly accidental mixing of academic or research work with a programming languages and its associated infrastructures is fortuitous. It allows us to transit between different strata of the social fields of science, engineering, health, medicine, business media and government more easily.

division of labour and possibly refined co-operative labour processes Chambers, John
operate around data.

Third, R is an increasing well-known and widely used statistical programming language and environment (R Development Core Team 2010). Its growth is perhaps just as important as its operation (Mackenzie 2014b). An open source programming language, according to surveys of business and scientific users, at the time of writing, R has replaced popular statistical software packages such as SPSS, SAS and Stata as the statistical and data analysis tool of choice for many people in business, government, and sciences ranging from political science to genomics, from quantitative finance to climatology (RexerAnalytics 2010). Developed in New Zealand in the mid-1990s, and like many open source software projects, emulating S, a commercialised predecessor developed at AT&T Bell Labs during the 1980s, R is now extremely widely used across life and physical sciences, as well as quantitative social sciences. John Chambers, the designer of S, was awarded the Association for Computing Machinery (ACM) ‘Software System Award’ in 1998 for ‘the S system, which has forever altered how people analyze, visualize, and manipulate data’ (ACM 2013). Many undergraduate and graduate students today earn R as a basic tool for statistics. Skills in R are often seen as essential pre-requisite for scientific researchers, especially in the life sciences. (In engineering, Matlab is widely used.) Research articles and textbooks in statistics commonly both use R to demonstrate methods and techniques, and create R packages to distribute the techniques and sample data. Nearly all of these publication-related software packages, including quite a few from the authors of *Elements of Statistical Learning* are soon or later available from the ‘Comprehensive R Archive Network (CRAN)’ (CRAN 2010). Estimates of its number of users range between 250000 and 2 million. Increasingly, R is integrated into commercial

services and products (for instance, SAS, a widely used business data analysis system now has an R interface; Norman Nie, one of the original developers of the SPSS package heavily used in social sciences, now leads a business, Revolution, devoted to commercialising R; R is heavily used at Google, at FaceBook, and by quantitative traders in hedge funds; in 2013 ‘R usage is sky-rocketing’; etc.). In general terms, R has a kind of disciplinary polyglot currency as a form of expression, and exhibits a fine-grained relationality with many different epistemic and operational situations associated with machine learning.

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

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

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

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

Nie, the founder of Revolution Analytics, a company that provides software and support services to R users, promotes R as a way of protect individual liberties to know what they

to use the psychoanalyst Christopher Bollas' term (Bollas 2008) , an object Bollas, Christopher through which many different ways of thinking circulate. Standing thinking somewhere at the intersection of statistics and computing, modelling and programming, many different disciplines, techniques, domains and actors intersect in R. Bollas suggests that 'our encounter, engagement with, and sometimes our employment of, actual things is a *way of thinking*' (92). R embodies something of plurality of practices of working with measurements, numbers, text, images, models and equations, with techniques for sampling and sorting, with probability distributions and random numbers. It engages immediately, practically and widely with words, numbers, images, symbols, signals, sensors, forms, instruments and above all abstract forms such as mathematical functions like probability distributions and many different architectural forms (vectors, matrices, arrays, etc.), as it employs data.

Writing code has always been central in machine learning where algorithms and machines are the primary expressive forms that ideas take as they

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

Ripley, Brian
Venables, Bill

become diagrams. Two of the main proponents of **R** and **S** describe the motivation for the language:

The goal of the **S** language ... is “to turn ideas into software, quickly and faithfully” ... it is the duty of the responsible data analysts to engage in this process ... the exercise of drafting an algorithm to the level of precision that programming requires can in itself clarify ideas and promote rigorous intellectual scrutiny. ... Turning ideas into software in this way need not be an unpleasant duty. (W. Venables and B. D. Ripley 2000, 2)

Bill Venables and Brian Ripley, statisticians working on developing **S**, the almost identical commercial predecessor to **R**, wrote in the early 1990s of the responsibility of data analysts to write not just use software. They write ‘software’ here not in the sense of a product, but in the sense that today would more likely be called ‘code.’ This sense of coding and programming as clarifying and concretising ideas with precision has thoroughly taken hold in contemporary data analysis. But not that it lies somewhat at odds with Pedro Domingos characterisation of machine learning as a shift away from building to growing. Many practices, many ways of doing things, describe themselves as part of their doing. Perhaps every practice contains elements of its own description. The contrast between **R** as a language originating in statistics and subsequently percolating through various scientific fields and **Python** or **Java**, which are much more general purpose programming languages widely used for software development in many settings, is instructive. Implementations of machine learning algorithms in **Python** can more or less be found in a few software libraries such as **SciPy** and **ScikitLearn** (Pedregosa et al. 2011), and much of the instructional materials on how to use machine learning in practice rely on **Python**. But these implementations are somewhat less

diagrammatically rich than those associated with R. They tends to be operational rather epistemic.

hyperobject!machine
learning as
table

The diagram of a hyperobject

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

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

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

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

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

How often suggested	
testthat	195
knitr	135
MASS	65
testthat, knitr	40
RUnit	38
parallel	29
lattice	25
knitr, testthat	20
ggplot2	16
survival	16
mvtnorm	15
R.rsp	13
rgl	12
testthat, roxygen2	10
xtable	8
akima	6
coda	6
scatterplot3d	6
tcltk	6
testthat, roxygen2, knitr	6

Table 2.4: R packages suggested by the ElemStatLearn package

Chapter 3

The vector space and its movements

Do machine learners want to go everywhere in the world, take on any problem, and see no data as too difficult to learn on? This formulation from Matthew Kirk's *Thoughtful Machine Learning* is typical:

Machine learning is an amazing application of computation because it tackles problems that are straight out of science fiction. These algorithms can solve voice recognition, mapping, recommendations, and disease detection. The applications are endless, which is what makes machine learning so fascinating
(Kirk 2014, 11)

It is harder but not impossible to find dissenting or sceptical voices. Computer scientists such as Anand Rajaraman and David Ullman in their *Mining Massive Datasets* voice doubts about the general power of machine learning algorithms and contrast these algorithms with specialized search and retrieval techniques (Rajaraman and Ullman 2012). Similarly, the

data!vectorisation
| see common vector space!vectorisation
vectorisation
data!volume

mathematician Cathy O’Neil trenchantly criticises the rapid extension of machine learning approaches in finance. Regardless of reservations and criticisms, the belief in ‘endless’ applications is hardly exaggerated. Machine learning has, once you start looking for it, become astonishingly generalized.

The epistemic mobility of its operation is not only due to the generic mathematical, statistical or computational advances. It attests to the advent of a new model of truth, which opens onto an expanding epistemic continuum. The continuum arises, I will suggest, from a very specific diagrammatization of data. This diagrammatic movement generates a continuum, a continuum that has already surfaced in the perceptron’s treatment of NAND truth table as discussed in chapter 2, that *vectorises* data. Machine learning moves data vectorally; it displaces data in specific directions.

Often data is represented as if it is an homogeneous mass or a continuous stream. If there is any such homogeneity, we should examine the transformations that allow different shapes and densities of data to mix. Data in its local complexes spaces out in many different density shapes, depending on how the data has been generated or instanced¹. Whatever the starting point – a measuring instrument, people clicking and typing on websites, a device like a camera, a random number generator, etc. – a given machine learner draws data into specific *vectorised* shapes (vectors, matrices, arrays, etc), and will scale it within a geometrically coordinate volume. The process of composing data for statistical, visual, predictive or even storage purposes, seeks to map a concrete situation, some state of affairs, onto forms imbued with various geometrical, probabilistic, and decisionist attributes that fit the mathematical functions or mathematical models. This

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.

mapping and forming is sometimes referred to as ‘data cleaning’ but this term covers over important but largely taken for granted transformations. If, as I have been proposing, we seek contact with the practices or concrete value-situations, then the reshaping and reflowing of data densities into vectors matters greatly. This forming and reforming of data is evidence of implicated relations. These practices attest to what 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 seek to elicit relations that strain the linear shaping of data, that show how it does not fit. Some practices working in the name of linear forms effectively trace strain feelings. These feelings are affective connectors between the abstractions and the concrete-value situations in which data arises.

Statistical modelling, data-mining, pattern recognition, recommendation systems, network modelling and machine learning rely very much on the transformation called ‘fitting a model.’ 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 continuum, which I term the *common vector space*. 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)). Vectoral movements into data tap into and are interwoven with the transformation of data along much more diagrammatic

machine learner!random forest lines.²

Breiman, Leo
statistics

In *Elements of Statistical Learning*, data appears in various forms. Plots of New Zealand fishing patterns lie next to plots of factors in South African heart disease. The 21 datasets shown in table ?? typify the variety of domains found in (Hastie, Tibshirani, and Friedman 2009, 2001). 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 span of situations is not unusual for machine learners. To give another example, the statistician Leo Breiman’s 2001 article on random forests (L. Breiman 2001) is one of the most cited journal papers in the machine learning literature and 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, Sat-images, Zip-code, Waveform, Twonorm, Threenorm, Ringnorm, (L. Breiman 2001, 12).

In some ways, the improbable conjunction of spam email and cancer detection in machine learning simply 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 this trekking movement). But in *Elements of Statistical Learning* and the field of machine learning more generally,

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.

something more is moving through and associating these datasets. The bagging colligation of datasets – vowels, ozone, bone density, marketing, prostate generalization cancer, and spam – in all their diversity demonstrates the mobility of machine learners. The diversity of the data could be seen as almost aleatory, as if the datasets were the fruit of random derivè in the world. Rather than random drifting, I think it more likely that the diversity of datasets allows machine learners to perform something like one of its own statistical procedures in ‘bagging’ (bootstrap aggregating (Hastie, Tibshirani, and Friedman 2009, 300)) datasets in order to average its predictions out across different domains. The repeated sampling of the world bootstraps machine learning as a field into its operational mode of generalization.

The tabular form, the practices of naming and labelling, and the sorting of different data types exemplified in these diverse datasets can tell us a lot about how machine learning organises and energises vectoral movement through data. The different shapes and composition of the datasets provide indications of the functioning of machine learners as they make knowledge or operationalise power relations. Datasets have visible forms of alignments or architectures that we should pay close attention to as they are learned machinically. Already in table ?? differences between datasets labelled ‘training’ and ‘test’ suggest something of this learning. We might also notice differences between data called ‘simulated’ and other kinds of data. If machine learning can be understood as a constantly evolving diagram or an abstracting multiplicity, the way it moves through data and associates different forms of data matter. This movement and connecting is somewhat guided by the shape of the data in some ways (for instance, very many demonstrations of machine learning use R.A. Fisher’s *iris* dataset (R. A. Fisher 1936)). But as we will see in this chapter, it also generates data in an increasingly extensive, heavily conventional space, the common vector

common vector
space!conventions of
referential!things and
activities
dataset!spam
dataset|zip

space. Building on the diagrammatic account of the *Elements of Statistical Learning* developed in the previous chapter, this chapter attends to the conventions, regularities and the mobilities of data in the common vector space.

Mixing media, medicine, business and biology

References to things act simultaneously as reference to (and within) activities. (Lynch 1993, 193)

‘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 (Hastie, Tibshirani, and Friedman 2009, 2), a table of spam email word frequencies appears (and the problem of spam classification is canonical in the machine learning literature - we return to it 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 ??). The final example in the introduction, ‘Example 4: DNA Expression Microarrays,’ draws this time from biology, and particularly, high-throughput genomic biology, the kind of science that produces large amounts of data about

something in the world by running many tests, or by constructing devices that generate many measurements, in this case, a DNA microarray.³ The image shown here is perhaps most striking. No numbers are shown, only a colorful heatmap with brighter colours standing for higher levels of gene expression, and darker colours for lower levels. For all its dense color, the data shown here is a sample of 100 of the approximately 7000 genes in the dataset `nci` (Hastie, Tibshirani, and Friedman 2009, 6). In comparison to the medical data from the `prostate` dataset with its 97 rows of 10 columns, the microarray dataset has 64 columns of data.⁴ Both are cancer datasets, but the `nci` dataset cannot be shown in its entirety because it refers to 60 different cell lines ranging across colon, breast, prostate, ovarian and renal cancers, as well as the thousands of different genes.

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 places these datasets come from. In passing, we should note that the *Elements of Statistical Learning* is not alone in this juxtapositional opening. Very similar example sets can be seen arrayed in most machine learning publications. To give just a few examples: Andrew Ng's CS229 lectures, for instance, treat spam in Lecture 5 ([Lecture 8 / Machine Learning \(Stanford\) 2008](#)); Rachel Schutt and Cathy O'Neil's *Doing Data Science* discusses spam in Chapter 4 (Schutt and O'Neil 2013), and Peter Flach's *Machine Learning The Art and Science of Algorithms that Make Sense of Data* (Flach 2012) introduces spam in the first few pages. Similar parallels exist around

3. This kind of 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.

4. The data derives from a publication in *nature genetics* (ross_2000) analysing gene expression in the cancer cell lines maintained as experimental models by the us national cancer institute.

table!history |(

the image recognition problem (exemplified by handwritten digits), around the measurements dataset (`prostate` in the case of (Hastie, Tibshirani, and Friedman 2009), and in relation to the larger, impossible-to-see-the-patterns datasets of the cancer microarray data). 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 are dealing with here produce, it seems, a *regularity* or a continuous common space, a space that displays strong tendencies to expand.

Truth is no longer in the table?

‘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 of course was writing 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. The ‘Classical’ table as a space of order was abandoned because it could no longer serve as the ‘common place for representation and things, for empirical visibility and for the essential rules’ (239). 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. These tables themselves supplanted earlier much more variegated Renaissance tabulations. 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 light of the 19th century, still want to separate. While the history of Linnaeus, Carl tables as data forms reaches a long way back (see (Marchese 2013) for a similarity broad historical overview that reaches back to Mesopotamia), and tables calculus still definitely abounded and indeed multiplied in the 19th and then 20th centuries, Foucault argues that the system of grids that permitted ranking, sorting and ordering in tables had between the end of the Renaissance and the early nineteenth been based on ‘buried similitudes’ and ‘invisible analogies’ (Foucault 1992 [1966], 26). The Classical Age grid or table was a way of bringing plural and diverse resemblances into exhaustive enumeration where they could be subjected to analysis by counting and comparison. The table as space of order did not stand in isolation. It served a particular epistemic function. While algebra or *mathesis* more generally applied to ‘simple natures’ (planets in movement, dynamics of falling bodies, etc.), table-based knowledges such as taxonomy dealt with more complex natures. Even in the tables, systems of signs – 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 the genesis of things (including grammar, nature and wealth) 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. This is an important limitation since it is precisely the complex natures in genesis that machine learning tries to engage with using algebra, calculus, statistics and computation.

statistics!history

In the nineteenth century, a different form of ordering shattered tabulation based on similarity and resemblance. Foucault figures this change as shattering:

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 – 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 (239-240).

Life, labour and language – the figure of the human – famously replace the exhaustive, aggregative, synoptic classificatory tables of the Classical age. Tables still abound in newer orderings (almost any episode from the history of nineteenth and twentieth century statistics will confirm that; see (Stigler 1986, 2002)) , but from now on there are things, organically structured, relating to a place and changing in time, and there are representations addressed to (and constitute) of a subjectivity. Double interiority takes over. Things such as a language, a species or an economic system have their own genesis, and our knowledges and indeed experience too become finite, historical, with their own dynamics and internal life. Knowledge is, for instance, epistemologized in various domains so as to become scientific

(through the use of experimental practice, certain ways of writing, etc.; table!history|) see (Shapin and Schaffer 1989) for one account of this change), but in this biopower change, the table itself is no longer the foundation of knowledge. It is one empiricities apparatus amongst many.

This brief résumé of one of the main arguments in *The Order of Things* might help us reassess what machine learning does in data. The question here is how we make sense of these tables in terms of their internal structures and their relation to what lies outside them. Presumably the `spam`, `prostate`, `image` and `microarray` data do not operate in the way Linnaeus would have put together a table of living things based on similarities and resemblances. But, as we will soon see, measures of similarity and resemblance operate strongly in machine learning as it moves through tables. In this respect, the Classical table perhaps returns with renewed relevance. Moreover, the repeated juxtaposition of tables of diverse provenance alongside each other suggests we should be looking for alignments and resemblances between whole tables as well as between particular cells, rows, columns or margins of a table. The topics of these tables – as we have seen, in the opening pages of *Elements of Statistical Learning* they concern work, life, language and economy in various ways – maps 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). Things may no longer be hedged in by the table. Their fundamental truth may have escaped from the space of similarities found in the Classical table. And yet we are now confronted by tables whose expansion and assimilation derives from calculations of similarity and distance. In many practical respects, we are hedged in by tables, and many different mechanisms animate and multiply tables around us.⁵ In certain respects the tables we see in the first

5. By and large, I am not discussing networking and database infrastructures here. In other work, I have attempted to account for the multiplication of tables in databases

pages of (Hastie, Tibshirani, and Friedman 2009) are much more classical in their scale and in their relatively immutability. In the flagship settings such as social media platforms or genomics in which machine learning operates, tables change rapidly in scale and sometimes in organisation. Regardless of any differences between the tables of data used to demonstrate machine learning techniques and data tables used in contemporary operational environments, the multiplication and juxtaposition of different tables today suggests that we might be seeing the advent of a non-classical common space of order for representation and things, for regularities and resemblances, for nature and imagination, or science and media. If that is so, then the question will be what kind of scanning, what kind of step-by-step, row-by-row movement, traverses them.

The epistopic fault line in tables

```
[r ElemStatLearn_data, echo=TRUE} library('ElemStatLearn')

datasets = data(package='ElemStatLearn') datasets[['results']][,3]
```

The `ElemStatLearn` R package brings, as we have seen in the previous chapter, with it around 20 different datasets, including the four we see in the opening examples. At this point, I'm not so much interested in gauging the rhetorical effect of this mixing of differences, as in exploring how *Elements of Statistical Learning* goes into and between data tables. On the one hand, every the data table indexes a localised complex of activities (clinical, social media platform, financial transactions, etc.). On the other hand, for machine learners, the table is a space of similarities and differences both within the table itself (e.g. how much does row number 1000 differ from

but also in mundane practices of ordering. See (Mackenzie 2011, 2012). My analysis there pivots mainly on a set-driven account of data derived from the work of Alain Badiou. Although I've already been using some set terminology here – as in ‘dataset’ – I’m persisting with a more geometrical and algebraic account of datasets in order to better deal with some of the graphisms common in machine learning.

row 1,000,000) and between tables (e.g. how much does this table of clinical test results differ from that table of microarray data?). These internal and external differences entwine with each other in ways that create a fault line. It is this fault or fold line, I propose, that diagrammatically transforms data tables into the expansive and moving substrate of the common vector space.

How can we see this fault line? In edging along this fault, we might proceed *epistemologically*. The term ‘epistemic’ comes from the work of the science studies scholar Mike Lynch , whose account of scientific practice is usefully focused on ordinariness. He proposed the ‘epistemic’ as a way of connecting ‘familiar themes from epistemology and general methodology’ (Lynch 1993, 280) with localized practices, or with the local achievement of coherence. In other words, as the term itself suggests, an epistemic 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-placedness is very useful in several ways, especially if it can be brought to bear on the problem of what happens when the ‘local complex’ of a specific dataset encounters a generalizing epistemic practice such as machine learning with its concerns with generalization, error, and prediction. Its interest in the couplings between ‘graphism’ and practice should help make sense of some of the diagrams (plots, equations, tables) we see in *Elements of Statistical Learning* as forms of movement that epistemologize data, or that inscribe thresholds of epistemologization in data. A focus on the connections between down-to-earth practices such as drawing a line, labelling a point or sorting a list of values and seemingly generic scientific-epistemological statements about error, bias, variation, confidence, expectation and likelihood is a good antidote to the sometimes ballistic epistemological trajectories fuelled by

diagrammatic movement
epistopic
Lynch, Mike
practice!scientific
machine
learning!statistical
aspects
epistemologiza-
tion!threshold
of|seealsoepistopic

epistemology statistical machine learning (e.g. the infamous and heavily cited comments
 Anderson, Chris by the former *Wired* magazine editor, Chris Anderson, on the ‘end of
 Bellman, Richard!curse theory’).⁶
 of dimensionality

common vector space!di-
 mensionality!curse
 of

Surface and depths: the problem of volume in data

Every distinct column in a table effectively adds a new dimension to the common vector space. Since the 1950s, problems of classification and prediction in high-dimensional dataspaces 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 dataspace 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 sparsely populated spaces because they accommodate many different boundaries.

6. 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 highly 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 (ng_2008a)

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, decompose, invert and generally transform matrices and vectors. They will be ‘highly optimised’ not because every programming language has been prepared for the advent of machine learning on a large scale, but rather more likely because matrix operations are just so widely used in image and audio processing. Happily, Ng observes, that means that ‘you can make lots of predictions’ (*Lecture 2 / Machine Learning (Stanford)* 2008). It seems that generating predictions and hypotheses outweighs the value of understanding how things work on this point.

In the epistopic mode of diagramming, machine learners find edges in data tables that are capable of combining or conjugating with each other, even if they are not reducible to each other. Let us see how Hastie, Tibshirani and Friedman begin to move through the datasets. First of all, they highlight in the first three (`spam`, `prostate` and `zip`, leaving aside the cancer genes dataset (`nci`)), what they have in common:

The first three examples described in Chapter 1 have several components in common. For each there is a set of variables that might be denoted as inputs, which are measured or preset. These have some influence on one or more outputs. For each example the goal is to use the inputs to predict the values of the outputs. This exercise is called supervised learning. We have used the more modern language of machine learning.(Hastie, Tibshirani, and Friedman 2009, 9).

In the ensuing discussion, they begin to dissect the tables and point to various discontinuities traversing the tables. The major discontinuity is already remarked in the common cut between ‘inputs which are measured’ and values that might called ‘outputs’ that inputs ‘influence.’ Not all tables are cut by machine learners in this way, but this cut is epistopic since the table now shows that some things are measured and some might not be (the values of the outputs). In addition to this vertical discontinuity dividing tables, more diverse horizontal discontinuities can be seen (and Hastie et. al. go on to describe it). Some of the variables in the table are continuous numerical values, others are discrete (they can only take on certain values such as a whole number), and indeed may be categorical or nominal (for instance, when predicting the actual digit from the handwritten digits, the predicted digit itself is a categorical variable, since the prediction must choose one of the digits 0,1,2,3,4,5,6,7,8 or 9). Discrete qualitative data

machine
learner!epistopic mode
data!variable!types

classification

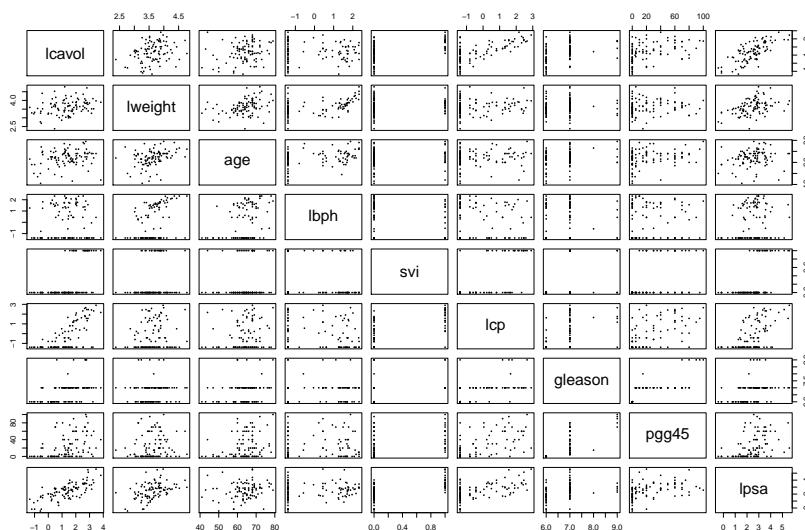


Figure 3.1: Scatter plot matrix of prostate data

appears frequently in machine learning, or wherever classification problems can be found. (I will have much more to say about classification and machine learning classifiers in later chapters. For the moment, we need only note that predicting the class to which things belong is a major topic of interest.)

```
library("ElemStatLearn")
data("prostate")
pairs(prostate[, -10], cex = 0.2)
```

Of the three example datasets (`prostate`, `spam` and `zip`), the authors return most frequently to `prostate`. This dataset derives from the work of urologists working at Stanford (Stamey et al. 1989), and, as (Hastie, Tibshirani, and Friedman 2009, 67) points out, 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. In the first pages of the book, they had already quite exhaustively plotted all the

variables in the dataset against each other using a scatter plot matrix (3) `dataset!prostate` (show in 3.1, and they return to the same data on almost a dozen occasions `graphics!scatterplot` in the course of the book.

`matrix`

`vector`

The transformation between the Table 3.2 and the Figure ?? already evinces a vector of movement commonly found around such datasets. On the one hand, the table arrays all the different data types indifferently in rows and columns. 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 squares 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-left to bottom right in figure ???. 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. Note how certain variables such as `svi` are not very amenable to plotting in this way. More importantly perhaps, note how certain combinations of variables result in visible forms can be seen as signs of relations between different variables. The matrix constructs a space in which contrasts between the pairs of variables that yield scattered clouds of points and those that show distinct clusters or tendencies can be seen. 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-textual grid. Everything remains on the surface of a grid here, but the grid functions differently in the scatterplot matrix.

All of this lies somewhat in a pre-machine learning space. Similar tables and plots are part and parcel of statistical data analysis more generally (see (Beniger and Robyn 1978) for an historical account of quantitative graphics in statistics statistics!graphics|seealso{graphics}). But the point of the scatterplot matrix is not to exhaust the dataset, but rather to highlight the need to constantly revisit it (12 times in the *Elements of Statistical Learning*) in order to tease out the multiple relations or influences that remain opaque to even the most exhaustive matrices of plots. We cannot clearly see in the scatterplot matrix beyond pairs of variables in relation. If the crucial diagnostic measure in this case is the level of the PSA (prostate specific antigen), how do we know what combinations of other measurements might be associated with its elevation? What is multiple variables affect the level of PSA?⁷ This question can 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 could be supported by the same figure. Perhaps worse, the very question of relation between multiple variables and the predicted levels of PSA suggests the existence of a hidden volume, an occluded or internal space that cannot be seen in a data table, or that cannot be brought to light in the geometry of a plot. This

7. 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 unbound points in the matrix (Stamey_2004)

volume is not the measured volume of the prostate but the virtual volume suggested by both the dataset table and the scatterplot matrix, a nine dimensional space subtended by treating each of the nine variables in the dataset as occupying a dimension. When Foucault wrote of truth escaping the table, he might well have pointed towards the higher-dimensional volumes of the *vector space* into which lines were already beginning to regress even in the late 18th century.⁸. A different basis of order – the common vector space – begins to take shape when the variables (usually columns) become dimensions. Strenuous efforts are made in machine learning to ensure that as much as possible goes into and comes of that common space.

common vector
space!dimension
common vector space|(classification

Vector spaces expand

To show this space in its epistopic making, we might follow what happens to just one or two columns of the `prostate` data in the common vector space as it is prepared for machine learning. In the `prostate` dataset, some variables are continuous quantitative values, and some are categorical (they represent membership in a group or category) or ordinal variables (they represent a ranking or order). But a variable such as `svi` has `True` or `False` values. How can such values be positioned in the vector space, and thus be subject to calculation in the same way as a measure of prostate gland volume? 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 so that it can be mobilised in the way that equation ?? (see chapter 2), the vector

8. 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

linear regression model form of the linear regression model, suggests it might be? The problem is data!variable|~~several~~~~data~~ variable of binary coding:
common vector space|)

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 common space permits once things inhabit it. Note that a single qualitative or categorical variable expands into ‘a vector of K binary variables or bits.’ The dimensions of the vector space expand accordingly, and the machine learners treat these added dimensions as variables to be included in the model. Qualitative data, once coded in this way, can be multiplied, added, and in short, handled algebraically using the same aggregate operations we saw in discussing linear algebra more generally. Note also that not only has the vector space expanded here, this expansion smooths over important gaps or differences that figure large in the dataset. 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.

Traversing behind the light

Adding dimensions to the common vector space makes seeing the volumes and densities of data distributed in this space more challenging. The character of the transformations in `prostate` that ensue in *Elements of Statistical Learning* are difficult to summarise. They are the locus of machine learning as an epistopic form of movement. In terms of the historical transformations of the table, they can be seen as subtending differences in a common vector space of simple natures understood as things with direction and displacement, and hence susceptible to calculation as vectors. Once this hidden virtual volume in the data is glimpsed, strenuous efforts will be made to bring it to light, even sometimes blatantly disregarding the local complex of activities that originally produced the data. These efforts will proceed along different lines. Sometimes this space is treated as one filled with constantly varying proximities and similarities, and machine learning techniques gather and order these differences (for instance, as in the k nearest neighbours model) or in unsupervised methods such as k-means clustering (513). More commonly, machine learning draws lines or flat surfaces that constrain through the volume using some version of linear regression, perhaps the most important modelling technique in modern statistics.

The importance of the linear modelling, finding lines of best fit, should not be under-estimated amidst the plethora of machine learners that attract more recent attention (neural nets, support vector machines, random forests). Linear regression lays down the basis of common vector space that 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.⁹ While it is more or less

machine
learning!epistopic
common vector
space!dimension
\texit{k-nearest
neighbours}model
\testit{k-means
clustering
linear regression
common vector
space|strain
linear model

9. One sign of the centrality of the line in machine learning can be seen, for instance,

linear algebra|()

obvious to the eye from the scatterplot matrix that lines could be drawn through the clouds of points as a way of defining directions of movement, the way to draw these lines in higher dimensions is not obvious. Yet the line of best fit has a readily graspable geometrical intuition to it, even in higher dimensions, and that line can be diagrammed by making use of the equations of linear algebra, the field of mathematics that operates on lines in spaces of arbitrary dimensions. It is no accident that linear algebra is such a taken-for-granted part of machine learning that its techniques for finding intersections between lines and planes, of manipulating collections of lines and surfaces through mappings and transformations (rotations, displacements or translations, skewing, and scaling), and above all, treating systems of equations using aggregate forms such as matrices and vectors. 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 resort to constantly.¹⁰

Many of these formalisations quickly become difficult to geometrically figure. We will have reason to examine some important ways in which linear algebra

from the contents page of the book ([hastie_2009](#)). 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](#)).

10. Along with statistics and probability, linear algebra is 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)

structures machine learning practices in later chapters, but for the moment, it might be understood as offering a way to draw lines through spaces that can only be expressed diagrammatically in the form of equations, not in the geometrical form of figures.¹¹ Let us return to the equations for linear regression models a third time (remembering that both C.S. Pierce and Andrew Ng advocate returning often to simple expressions). The ‘mainstay of statistics,’ the linear regression model, usually appears diagrammatically in the form of linear algebra:

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

$$\hat{Y} = X_T \hat{\beta} \quad (3.2)$$

The concision of this way of diagramming the drawing of line through a high

11. We might add also approach the epistemic 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 1988), 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 others. In ‘The Becoming Topological of Culture,’ Lury, Luciana Parisi and Tiziana Terranova suggests that ‘a new rationality is emerging: the moving ratio of a topological culture’ (Lury, Parisi, and Terranova 2012, 4) {Lury, Celia}. In this new rationality, practices of ordering, modelling, networking and mapping co-constitute culture, technology and science (5). At the core of this new rationality, however, lies a new ordering of continuity. The ‘ordering of continuity,’ Lury, Parisi and Terranova propose, takes shape ‘in practices of sorting, naming, numbering, comparing, listing, and calculating’ (4). The phrase ‘ordering of continuity’ is interesting, since we don’t normally think of continuities as subject to ordering. In many ways, that which is continuous bears within it its own ordering, its own immanent seriation or lamination. But in the becoming topological of culture, movement itself undergoes a transformation according to these authors. Rather than movement as something moving from place to place relatively unchanged (as in geometrical translation), movement should be understood as more like an animation, a set of shape-changing operations. These transformations, I would suggest, should be legible in the way that machine learning, almost the epitome of the processes of modelling and calculation that Lury, Parisi and Terranova point to, itself moves through the data. And indeed, the juxtaposition of spam, biomedical data, gene expression data and handwritten digits already suggests that topological equivalences, and a ‘radical expansion’ of comparison might be occurring. Bringing epistemics and topologies together might, I suggest, help trace, map and importantly diagram some of the movements into the data occurring today.

dataset!prostate dimensional space derives largely from linear algebra. Reading Equation
 deep learning **8.2** from left to right, the expression \hat{Y} already points to a set of calculated,
 parameters|seealso{model!coefficient} predicted values, or a vector of y values, such as all the `lpsa` or PSA
 model!coefficient 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’ (11). 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).

So, in the expression for Equation ?? we can begin to grasp the diagramming
 of a line that cannot be fully drawn in any figure, only projected onto
 the dimensions of a plot. In spite of these limitations, it can be readily
 imagined and conceptualised within the expandable dimensions of vector
 spaces. While that line can never fully come to light, the diagrammatics
 of equations **8.2** and **3.2** expresses a way of constructing it calculating the
 coefficients $\hat{\beta}$. Much effort, many techniques, and sub-fields of science attach
 themselves to various ways of constructing and making statements about
 such lines. The problem of calculating optimal values of the coefficients
 β has attracted and continues to attract the attention of statisticians for
 a long time (at least two centuries) and today lies at the heart of cutting
 machine learners such as deep learning neural nets. 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 common vector space. This diagrammatic element is not necessarily reductive, although it does render differences as distances in vector space. Between the statement of the data and the system of coefficients defining a fitted surface, the diagrammatic expression of the linear model creates a new kind of anamorphic or diagonal space, ‘constituting hundreds of points of emergence or creativity, unexpected conjunctions or improbable continuums,’ as Deleuze puts it in describing such diagrams (Deleuze 1988, 35).

common vector
space|distance

Deleuze, Gilles!diagrams
common vector
space!linear
algebra|seealso linear
algebra

linear algebra|)

Drawing lines in a common space of transformation

The common vector space generated by the epistopic operation of machine learners has several facets. First of all, it invites mathematical analysis in terms of matrix operations and linear algebra . In high school mathematics, at least since WWII, students have been taught how to solve systems of linear equations, first using algebra, and then using matrix operations. Solving a system of linear equations means finding those values of the variables that satisfy all the equations in the system. If such values can be found, we know that the lines expressed by equations of the form $y = ax_1 + bx_2 + c$ intersect at a point, along a line, in some sub-space. In all of these cases, diligent mathematics students solve to find the common space (shown in figure 3.2), even if it consists in a single point, inhabited by the elements of the system.

Similarly, although the mathematics is slightly more complicated than high school level (although not by much), drawing the line of best fit through a set of data points can be seen as solving a system of linear equations. Viewed in terms of linear algebra, the ‘solution’ – or ‘closed form solution’ – to the linear model is given in equation 3.3:

ordinary least squares
calculus!differential

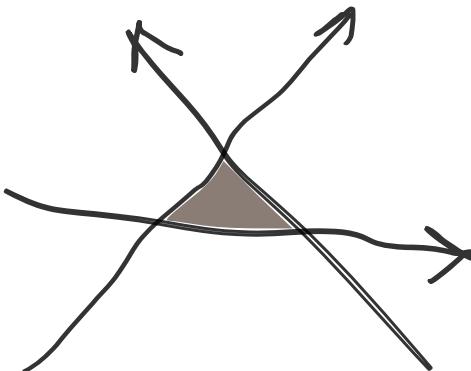


Figure 3.2: Common space in linear equations

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

In this expression, the data shown as \mathbf{X} supports the calculation of the coefficients $\hat{\beta}$ that define a flat surface or plane cutting across the common vector space.¹² 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. (As usual, and in keeping the diagrammatic reading of these forms, these operations all consist of ways of either moving or combining numbers in practice.) The relevant

12. 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 epistemic aspect of these lines remains prominent. Typically, many different statistical tests (Z-scores or standard errors, F-tests, confidence intervals, and then prediction errors) will be applied to any estimate of the coefficients of even the basic linear regression model, well before most advanced or sophisticated models and techniques (cross-validation, bootstrap testing, subset and shrinkage selection) begin to re-configure the model in more radical ways.

point is that equation 3.3 calculates some values for the linear regression model coefficients $\hat{\beta}$ and therefore assigns a slope and intercept, a direction and displacement to a line or flat surface traversing the density-shape of a dataset in its full dimensional vector space (nine dimensions in the case of prostate).¹³

-ply: vectorisation and the transformation of common space

The common vector space appears and operates just as directly in code, especially in R and other programming languages designed for data practice (Octave, Matlab, Python’s NumPy, or C++ Armadillo). *Vectorised* transformations of data lie are the moving substrate of machine learning as it expands. As usual, R coding practice instantiates this in miniature. As a programming language, R is striking for its many *vectorised* constructs and operations. 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’ (Teator 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. Or, as *R in a Nutshell* puts the point: ‘in R, any number that you enter ... is interpreted as a vector’ or as ‘an ordered collection of numbers’ (Adler and Beyer 2010,

13. As we will see in the following chapter (chapter 4), it is not always possible to calculate the parameters of a model analytically. Especially in relation to contemporary datasets that have very many variables and many instances (rows in the table), linear algebra approaches become unwieldy in their attempt to produce exact results, and machine learning steps in with a variety of computational optimisation techniques.

common vector
space!vector

17). In these operations, all data is a vector. That is, all data operations treat data as a movement with direction and magnitude. This implicitly links data to a spatial dimension, even if it has no obvious spatial index or reference. There are many vectorised operations in the R core language and many to be found in packages (the popular ‘plyr’ package; versions of ‘ply’ can also be found in recent Python data analysis libraries such as `numpy` or `pandas` (McKinney 2012)). In many cases, these vectorised operations occur implicitly. 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.

```
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
```

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

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

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

Church, Alonzo

programming!functional

The practical difference between the two approaches to moving through data is illustrated in the code listing ?? above 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, but when millions of numbers are handled, these implementation differences significantly affect the work of both programmers and computing machinery. 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. More importantly, this is a different mode of movement. Operations now longer step through a series of coordinates that address data elements, but wield plans, surfaces, cross-sections or stratifications of the vector space.

A further level of vectorisation appears in the vectorisation of functions in R. Specific R constructs such as `apply`, `sapply`, `tapply`, `lapply`, `mapply` exemplify these vectorised functions. All of the `-ply` constructs have a common feature: they take some collection of things (it may be ordered in many different ways – as a list, as a table, as an array, etc.), do something to it, and return something else. This hardly sounds startling. But while most programming languages in common use offer constructs to help deal with collections of things sequentially (for instance, by accessing each element of a list in turn and doing something with it), R offers ways of expressing a simultaneous operation on them all. The `-ply` constructs ultimately derive from the functional logic developed by the mathematician Alonzo Church in the 1930s (Church 1936, 1996). The functional programming style of applying functions to functions seems strangely abstract. These

cloud computing -ply constructs and the implicit vectorisation of many operations on data structures can be written concisely and tersely. They transform data in ways that readily adapt to and indeed motivate increasingly parallel contemporary chip architectures, clusters of computers, reallocation of computation to GPU (Graphic Processing Unit), FPGA (Field Programmable Gate Arrays) and, perhaps most significantly, the increasingly Cyclopean infrastructures of cloud computing and associated data centres. Many of these condensing and expanding movements of data are diagrammed in miniature in the R constructs as operators in vector space. The -ply operations reduce both data and computational frictions. The real stake in -plying data, is not speed but transformation. -plying makes working with data less like iteration (number, text, table, list, etc.), and more like folding a pliable material. Such shifts in feeling for data are very mundane yet crucial to the epistemic movements in data. While the examples of vectorised computation that I have shown are relatively trivial – adding sequences of numbers – these vectorised operations exert both infrastructural and epistemic strains for the common vector space.

The infrastructural implication we have just seen: the implicit and explicit vectorisation of data and operations feeds directly into the scaling up of computational work on data. Not only vectorised computation, but the various styles of programming and infrastructure that lend themselves to simultaneous transformations of large matrices or tables of data lie at the kernel of not only machine learning, but of the intensive processing of data in many different places. The rendering of moving images in digital video and in computer game animation depends implicitly on vectorised transformations of matrices of numbers (Mackenzie 2010). The more abstract implications of vectorisation and the forms of movement it encourages and proliferates bring us back to the problem of what it means for machine learning to learn.

. In many respects, it entails drawing a line through the data distributed in the common vector space. In short, vectorising computation makes the common vector space, which we might understand as a resurgent form of the table, operationally concrete and machinically abstract. It is no longer a formal diagram, but a machinic process that multiplies and propagates into the world along many diagonal lines.

machine
learning!learning
common vector
space!dimensionality

‘We fit a linear model’

‘We fit a linear model’ write Hastie and co-authors, referring to the `prostate` data. Models constrain movement in the fluxing dimensionality of common vector space. In R this might look like the code excerpt shown below:

```
library(ElemStatLearn)
library(xtable)
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])
tab1 = xtable(summary(prostate_model), caption = "Fitting a linear model", label = "tab:pros",
              type = "latex")
print(tab1, include.rownames = FALSE)
```

The epistopic character of Table 3.3 surfaces around the coefficients $\hat{\beta}$ that define the direction of a flat surface running through the common vector

```
dataset!prostate
common vector
space!vectorisation
```

space of the `prostate` data.¹⁴ This vector is a product of operations in that space. As Hastie and co-authors write, the ‘unique solution’ to the problem of fitting a linear model to a given dataset using the most 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 parameters $\hat{\beta}$ for a linear model. The two matrices involved are the X input variables (all of the `prostate` variables apart from the variable chosen as the response variable, in this case `lpsa`, the log of the PSA level) and a 1-dimensional matrix y , the `lpsa` values. These matrices are multiplied, transposed (a form of rotation that swaps rows for columns) and inverted (a more complex operation that finds another matrix). Epitomising the vectorised code often used in machine learning, calculating $\hat{\beta}$ for the `prostate` data only requires one line of R code:

The implicit vectorisation of the R code in the code listing ??, the fact that it already concretely operates in the common vector space, operationalizes the concise diagrammatizing of equation 3.3 as a machine process. More importantly, the vector of values that result from the vectorised multiplica-

14. 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 table of coefficients (see Table 3.3) and some new numbers, *statistics*. This table is not as extensive as the original data, the 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 a line has been fitted by the linear model to the data. The columns of the table now bear abbreviated and much more statistical names such as `estimate` (the estimated values of $\hat{\beta}$, the key parameters in any linear model), `Std. Error`, `t value`, and the all important p values written as `Pr(|t|)`. The numerical values ranging along the rows mostly range from -1 to 1, but the final column includes values that are incredibly small: 1.47e-06 is a few millionths. Other statistics range around the outside the table: the F-statistic, the R-squared statistic, and the Residual standard error. I don’t propose discussing these in any great detail here. No matter how we understand these statistics, they constitute a transformation of the `prostate` dataset. The numbers of the 3.2 become epistopic here, since they now appear as a set of standard errors, estimates, t-statistics, and p values, that together indicate how likely the estimated values of β are, and therefore how well the diagonal line expresses the relations between different dimensions of the dataset in the common vector space. How have these statistics, which all act as qualifications and qualifications on the line whose direction and point of entry is given the $\hat{\beta}$ coefficients, arisen? We seem to have crossed some threshold between drawing a line through an occluded data volume and generating propositions about the way that line passes through the data.

tion, transposition and inversion of matrices creates the new vector defined by $\hat{\beta}$ whose estimated values will be subjected to the statistical tests of Foucault, Michel significance, variation, and error that we see in Table 3.3. We will have occasion to return to these statistical estimates (in chapter 5), since the play of values that starts to appear even in just fitting one model will become much more significant when fitting hundreds or thousands of models, as some machine learners do.¹⁵

The vectorised table?

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

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.’ A version of this generative function appears in the machine learning shown by equations 8.2 and 3.3 and in the code listing ?. In the construction of the common vector space in linear algebra, and then its practice in code rather than solely in the analytical space of equations,

15. 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 derives often from the multiplication or indeed proliferation of models. Techniques such as neural networks, cross-validation, bagging, shrinkage and subset selection, and random forests, to name a few, generate many statistics, and navigating the multiple or highly variable models that result becomes a major concern. An epistemic abundance will appear here – bias, variance, precision, recall, training error, test error, expectation, Bayesian Information Criteria, etc. as well as graphisms such as ROC (Receiver-Operator-Characteristics) curves. Put simply, the proliferation of models start to drive the dimensional expansion of the common vector space. At the same time, the multiplicity of models multiplied by the machine learners becomes the topic of statistical analysis.

table
common vector
space!metatable

machine learning vectorises new volumes or dimensionalities that could not be seen, or rendered in either tables of data or graphic figures of data. Every machine learner inhabits and moves through the common vector space. Sometimes their operations flatten the vector space down into lower dimensions, sometimes they expand the vector space into a great many new dimensions (as we saw with ‘dummy variables’ that embody categories, and as we will see with support vector machine classifiers in a chapter 6).

The generalized application of machine learning relies on and reinforces a broad, powerful yet quite subtle transformation of data into vectors and a vector space. It remaps the grid of the table into the expanding dimensions of the common vector space. 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 2013b). But perhaps in the vectorised and matrix-form practices of the common vector space, machine learners produce for the first time a quasi-tabular or meta(s)table volume that cannot be surfaced on a page or screen, yet is heavily diagrammatically traversed through vectorised operations.

Does machine learning learn from the data? It certainly puts learning into the data. The epistemic transformation of the table pulls and re-aligns communication and infrastructures. It acts as a powerful tensor on knowledges and operations of many different kinds since it transposes, inverts, and remaps local complexes of activity. In following what happens to vectors, lists, matrices, arrays, dictionaries, sets, dataframes, or 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 fold and refold matrices and vectors in a common vector space. What is at stake in the common vector space? The broad stake, at the risk of over-emphasising it, is the production of new kinds

of realities occasioned by the mobility of machine learners. The machine learner recognising zip codes might be found in the navigation system of an autonomous vehicle or the methods section of a paper published in *Nature*. The datasets that *Elements of Statistical Learning* ranges sides by side evince this improbable diagonal movement.

The challenge lies in attending to this expansive data vectorisation in ways that maintain and indeed augment its value-relevance, its concreteness, and attachments to lives and places. We lack good intuitions of how to do that because of the ways in which data as vector space has been constructed and described. Does the act of unwinding some of these operations, for instance, by seeing how matrix multiplication ripples through different treatments of data in a linear regression model (even such a small one as the `prostate` data), bring us closer to the vitality of the multiple/multiplying concrete value-situations that connect calculation? My suggestion is that 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. As we will see, the vectoral operations we have just been viewing are themselves organised by other lines of diagrammatic movement that shape surfaces in more convoluted forms.

machine learner!mobility
of
machine
learning!epistopic
diagrammatic movement

	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*

	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

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

machine learners\variety
of|(
Domingos, Pedro

Chapter 4

Machines finding functions

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

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

Kirk, Matthew

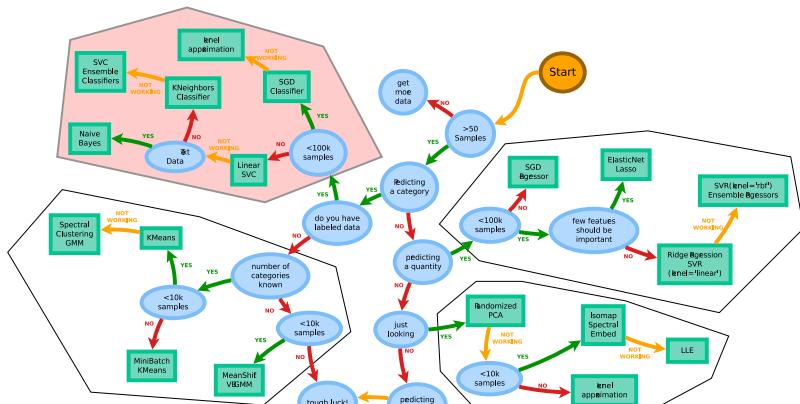


Figure 4.1: ‘scikit-learn’ map of machine learning techniques [TBA: ref to diagram]

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 weedlike abundance of machine learners, as does the 700 or so pages of *Elements of Statistical Learning*. Much learning of machine learning work, at least for many practitioners, concerns not so much implementation of particular techniques (neural network, decision tree, support vector machine, logistic regression, etc.), but rather navigating the maze of methods and variations that might be relevant to a particular situation.

How does this effect of accumulation and profusion arise? And what do machine learners do about it? The publications I have just been referring to present that profusion as a problem of the piling up of scientific publications. As the authors of textbooks and how-to-manuals, they attempt to manage it by providing, indexes, maps and guides to the bewildering variety of machine

learners. *Elements of Statistical Learning* deploys tables, overviews, theories of statistical modelling, model assessment and comparison techniques to aid in navigating them. Parallel and complementary mappings accompany software libraries. The visual map of machine learning techniques shown in Figure 4.1 comes from a particular software library written in Python, **scikit-learn** (Pedregosa et al. 2011). This software library is widely used in industry, research and commerce. In contrast to the pedagogical expositions, theoretical accounts or guides to reference implementation, code libraries such as **scikit-learn** tend to order the range of techniques by offering recipes and maps for the use of the *functions* the libraries supply. The branches in the figure lay down paths through the profusion of techniques as a decision tree.¹ Similarly, for R code, the *Comprehensive R Archive Network* tabulates key libraries of R code in a machine learning ‘task view’ (Hothorn 2014).

Python!scikit-learn
library
function
machine learners!variety
of()
programming
language!R!Compre-
hensive R Archive
Network

```
import sklearn

from sklearn import *
modules = dir(sklearn)

modules_clean = [m for m in modules if not m.startswith('_')]

print('\n'.join(modules_clean))
```

/usr/local/lib/python2.7/dist-packages/sklearn/pls.py:7: DeprecationWarning: This module has been moved to cross_decomposition and will be removed in 0.16 “removed in 0.16”, DeprecationWarning) base clone cluster covariance cross_decomposition cross_validation datasets decomposition dummy ensemble externals feature_extraction feature_selection gaussian_process grid_search hmm isotonic kernel_approximation lda learning_curve linear_model manifold metrics mixture multiclass naive_bayes

1. See Chapter 6 for discussion of decision trees in machine learning.

abstraction!levels of function!different senses of!mathematical, algorithmic, operational

neighbors neural_network pipeline pls preprocessing qda random_projection re semi_supervised setup_module svm sys test tree utils warnings

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

All three senses of function constantly coalesce and hybridize in machine learning research. Table 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, dominate here. Machine learners ‘find’, ‘estimate,’ ‘approximate,’ ‘analyse’ and sometimes ‘decompose’ mathematical functions. The primary mathematical sense of a function refers to a relation between sets of values or variables. (A variable is a symbol that can stand for a set of numbers or other values.) A function is one-to-one relation between two sets of values. It maps a set

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.

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 would 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). cost function!see{function!cost} The latter group I discuss below because they underpin many claims that machine learners learn.

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

common vector space

machine

learning!learning|()

Supervised or unsupervised, who learns what?

The techniques, models, forms of abstraction, data formats, and performance properties of machine learners have to be learned by people (reading books, attending classes, watching demonstrations, trying out software and code, etc.; see Chapter 8). The `scikit-learn` map of techniques shown in Figure 4.1 addresses the problem of choosing a machine learner. This map is only a starting point. No matter how powerful machine learners become, they do not operate autonomously. Once learned, maps such as the one shown in Figure 4.1 may become redundant. But other forms of close attention, monitoring, and observation remain crucial whenever machine learners encounter data or wherever they enter the common vector space.

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

Writing around 2000, Hastie et. al. state:

With supervised learning there is a clear measure of success or lack thereof, that can be used to judge adequacy in particular situations and to compare the effectiveness of different methods over various situations. Lack of success is directly measured by expected loss over the joint distribution $Pr(X, Y)$. This can be estimated in a variety of ways including cross-validation. In

the context of unsupervised learning, there is no such direct measure of success. ... This uncomfortable situation has led to heavy proliferation of proposed methods, since effectiveness is a matter of opinion and cannot be verified directly. (Hastie, Tibshirani, and Friedman 2009, 486-7)

data!training
data!test
partial
observer seefunction!as
partial observer

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

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

machine learning!learning|) function!mathematical|(a clear 'measure of success,' that success only seems to encourage further variations and comparisons that end up proliferating machine learners, their publications and their software implementations. Levels of predictive success may vary widely with different techniques and different situations, but an almost unbounded optimism associated with machine learning (for instance as more or less unspoken foundation of any analysis of 'big data') runs pell-mell across all of them.

Which function operates?

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

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

Writing in the 1980s, the statistician Leo Breiman describes classifiers – perhaps the key technical achievement of machine learning and certainly the catalyst of many applications of machine learning – in terms of functions. A classifier *is* a function $d(\mathbf{x})$ where \mathbf{x} is the data and d ranges over numbers that map onto categories, rankings or other forms of order and

belonging. The equation of machine learners to functions is quite pervasive. Breiman, Leo Learning, predictions, and the classifications produced by machine learning Vapnik, Vladimir derive from functions. The identification of machine learning with functions appears in the first pages of most machine learning textbooks. Learning in machine learning means finding a function that can identify or predict patterns in the data. As *Elements of Statistical Learning* puts it,

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

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

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

function!mathematical|) 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 admissible functions that minimizes the probability of error’ (Vapnik 1999, 31). Note the unobtrusive but crucial caveats in this formulation: functions must be found from a set of ‘admissible functions.’ Functions anchor machine learning so that we don’t have to come up with algorithms, yet functions themselves have to be found under certain constraints.

Which functions does machine learning admit? As is often the case in working with a massive technical literature, the first problem in making sense of what is happening with function in machine learning concerns sheer abundance. The pages of (Hastie, Tibshirani, and Friedman 2009) are marked with score of references to ‘functions’: quadratic function, likelihood function, sigmoid function, loss function, regression function, basis function, activation function, penalty functions, additive functions, kernel functions, step function, error function, constraint function, discriminant function, probability density function, weight function, coordinate function, neighborhood function, and the list goes on. This list stands against a background of several hundred mathematical functions commonly used in science and engineering.⁵ Clearly we cannot expect to understand the

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.

functioning of all these functions in any great detail. However, even a glance through this prickly list of terms begins to confirm the heavy reliance on functions in this field (as perhaps in many other science and engineering disciplines), but that the association between functions might be a way to map some important operations occurring in and around machine learning. We can also already see in this list that the qualifiers of the term function are diverse. Sometimes, the qualifier refers to a mathematical form – ‘quadratic,’ ‘coordinate’, ‘basis’ or ‘kernel’; sometimes it refers to statistical considerations – ‘likelihood’, ‘regression’, ‘error,’ or ‘probability density’; and sometimes it refers to some other concern that might relate to a particular modelling device or diagram – ‘activation,’ ‘weight’, ‘loss,’ ‘constraint,’ or ‘discriminant.’

What does a function learn?

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

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

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

scienceexperiment

in particular.⁶ But elsewhere 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 claims made about the functions in order to highlight the specific power of science: ‘celebrating the exceptional character of the experimental achievement very effectively limits the claims made in the name of science’ (Stengers 2011, 376). (Limiting claims made for science might save it from being totally re-purposed as a techno-economic innovation system.)

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

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

The generic term ‘reference’ here harbours a multitude of relations. The experimental achievement, the distinctive power of science, works through a tissue of relations that connect people, things, facts and mathematical

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. (Parisi 2013); see also my (Mackenzie 1997)), but Totaro and Ninno’s suggest that algorithmic processes, as the social practice of the function, form contradictory hybrids with remnants of substance, in particular, categories and classification. . They see bureaucratic logic, for instance, as hopelessly vitiated by a contradiction between classification and function. Machine learning, I’d suggest, is an important counter-example. It hybridises function and classification without any obvious contradiction.

functions in a highly heterogeneous weave.⁷ When a biomedical experiment uses *logistic regression* to ‘estimate the probability that a critically-ill lupus patient will not survive the first 72 hours of an initial emergency hospital visit’ (Malley, Malley, and Pajevic 2011, 5), they are doing machine learning, and the value of their predictions is not captured by classical statistical approaches (analysis of variance, correlations, regression analysis, etc.). As machine learning techniques and the underpinning mathematics of probabilistic learning theory circulate more widely across different scientific disciplines (geography, ecology, astronomy, epidemiology, genomics, chemistry, communication engineering), in each setting the experimental achievement consists in constructing references between the mathematical functions and the matters of fact generated by instruments, observations and measurements and cantilevered by previous experiments. The question from Stengers’ standpoint is this: what happens to the structure of referrals through experiments and the accumulated knowledge when functions are said to learn? In order to address this question, we need to delineate how functions function in machine learning. That could be in several different ways: as statements or utterances, as formula-diagrams, as graphic forms and in operational implementations as code. Any account of machine learning as function finding needs to map the concatenation of these different elements, none of which alone can anchor the ‘learning’ that goes on in machine learning.

At first glance, machine learning as a field is not very experimental (even it radically influences the conduct of experiments in many scientific fields; see chapter 7). It lacks the apparatus, the instruments, the laboratories, field sites or clinics of experimental practice. Experimentation, if there is any, takes place principally in the form of rendering diagrammatically visible

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

diagrammatic!experiment
function!mathematical
callinvention
of

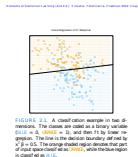


FIGURE 4.2a. A linear regression classifier has learned to separate the two classes. The classes are coded as a binary variable, orange = 1 and blue = 0. The line is the decision boundary, and the regions of fixed space closest to each class are shaded.

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

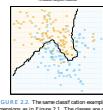


FIGURE 4.2b. The same linear classifier as in Figure 4.2a, but with a wavy decision boundary learned by 15-nearest neighbor averaging as in (2.8). The prediction is made by majority vote amongst the 15-nearest neighbors.

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

the relays or referrals between different functions as they traverse data. While functions are often written in formulae, they are also diagrammed in graphic forms as plots and in operational forms as code or software. Many of the characteristic functions listed above appear in the graphic form of lines and curves. This diagrammatic entanglement of machine learners in curves is not surprising. The historical invention of the term ‘function’ by the philosopher G.W. Leibniz in the 17th century relates to curves and their description. Functions for Leibniz describe variations in curves such as their slope. Identifying and locating these *singularities* in curves still preoccupies the function-finding done in machine learning. In contrast to the table, or the generalized common vector space that expands to accommodate all differences, the curve and the many graphic objects that seek to show curves in different ways, display continuous variation and singular points.

The 15-nearest neighbour classifier shown in Figure 4.2b layers simulated data (2 dimensions and an output variable with the values **BLUE** or **ORANGE** (Hastie, Tibshirani, and Friedman 2009, 12)) and a meandering line that classifies the simulated observations in terms of their class membership.

Viewed in terms of their visual composition, many machine learning diagrams are organised around such lines that contour, divide, bound or surround the data captured from or emitted by the world. Curving lines outnumber all other graphic forms, whether the curves are the ‘decision boundaries,’ the plots of ‘training errors,’ the ‘contours of constant density’ (109), or the ‘regularization path’ for the South African heart data (126). There is a constant tension in these graphic forms between line and curve. Consonant with the vectoral ideal of a straight line that either connects or cuts the data (as shown in Figure 4.2a), many of the graphics in the book (and others like it, especially the books written by computer scientists) show strong preferences for straight lines. But variations and uncertainties of various kinds detour the pursuit of the straight lines. Curves proliferate as machine learners try to soften the rigidity of the lines or find regular paths for lines through irregular terrain. Viewed very naively, the contrast between lines and curves in the two figures above, a contrast replayed many times in *Elements of Statistical Learning*, suggests that different diagrammatic operations are at work around the data, sometimes aligning and straightening relations (for instance, the many linear models) and sometimes tracing much more non-linear paths through the data (for instance, as in k nearest neighbours or much more indirectly convolutional neural networks). In the several hundred colour graphic plots in (Hastie, Tibshirani, and Friedman 2009), a striking mixture of network diagrams, scatterplots, barcharts, histograms, heatmaps, boxplots, maps, contour plots, dendograms and 3D plots respond to this tension between linearity and curvature.⁸ Many of these graphic forms are common in statistics (histograms and boxplots), but some relate specifically to data mining and statistical learning (for instance, ROC – Receiver Operating Curve – or regularization path plots). A significant proportion of

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

function!diagram
 common vector space
 function!sigmoid|
 seealso|function!logistic|
 a function|
 the two are almost synonymous) that refers the different data elements in the common vector space to each other.

Diagramming with curves: the logistic function

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

$$f(x) = 1/(1 + e^{-kx}) \quad (4.1)$$

The logistic function (shown as Equation 4.1 and as two curves in Figure 4.3), as we will see, is very important in many classification and decision settings precisely because of its *non_linear* shape and its constrained movement within a limited range of values (0 to 1). How does a function such as the sigmoid function ‘fit’ anything? Here the curve itself and even the name ‘sigmoid’ is the best guide. The S-shape of the sigmoid curve is

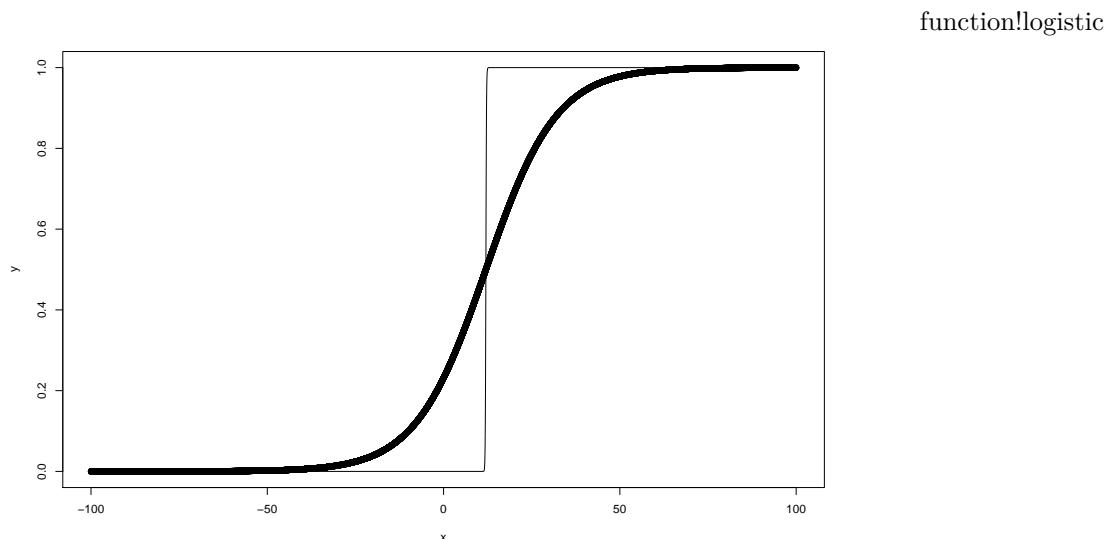


Figure 4.3: Logistic or sigmoid function

a good guide to operations associated with curves. The logistic function has quite a long history in statistics since that curve diagrams growth and change in various ways. (As the historian of statistics J.S. Cramer writes: ‘The logistic function was invented in the nineteenth century for the description of the growth of organisms and populations and for the course of autocatalytic chemical reactions’ (Cramer 2004, 614).⁹ In nearly all of these cases, the function was used to fit a curve to data on the growth of something: populations, reactions, tumours, tadpoles tails, oats and embryos. The reference of the curve to growth comes from its changing slope. Growth starts slowly, increases rapidly and then slows down again as it reaches a limit. In the second half of the twentieth century, it was widely used in economics. In all these settings and usages, the curve was a way of summarising and predicting growth. Census data, clinical or laboratory measurements supplied the actual values of $f(x)$ at particular times, the

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

function!parameters of x values. The task of the demographer, physiologist or economist was to function!referentiality!see also [nonlinearity](#) values of parameters such as k that controlled the shape of the curve. The logistic function had a well-established biopolitical resonance.

Note that the curves showing in Figure ?? plot the same data (\mathbf{X} and y values), but differ in their curvature. This diagrammatic variation derives from the parameter k , which discreetly appears in the equation 4.1 next to x . Such parameters are vital control points in function fitting and any learning associated with that. Varying these parameters and optimising their values is the basis of ‘useful approximation’ in machine learning. Sometimes these parameters can be varied so much as to suggest entirely different functions. In 4.3 for instance, $k = 12$ produces a much sharper curve, a curve that actually looks more like a qualitative change, range than a smooth transition from 0 to 1. The sharp shape of the logistic function when the scaling parameter k is larger suggests another important transformation, somewhat orthogonal to the description of rates of growth under limits. The mathematical function $f(x) = 1/(1 + e^{-x})$ can be treated as a way of mapping continuously varying numbers (the x values) and discrete values. Because $f(x)$ tends very quickly to converge on values of 1 or 0, it can be coded as ‘yes’/‘no’; ‘survived/deceased’, or any other binary difference. The transformation between the x values sliding continuously and the binary difference, classification or categorisation pivots on the combination of the exponential function (e^{-x}), which rapidly tends towards zero as x increases and rapidly tends towards inf as x decreases, and the $1/(1 + \dots)$, which converts high value denominators to almost zero, and low value denominators to one. This constrained path between variations in x and their mapping to the value of the function $f(x)$ is mathematically elementary, but typical of the relaying of references that allows functions to intersect with and constitute matters of fact and states of affairs. This

realisation – that a continuously varying sigmoid function could also map discrete outcomes – forms the basis of many machine learning classifiers. So, a contemporary biostatistical machine learning textbook can write: we can use logistic regression to ‘estimate the probability that a critically-ill lupus patient will not survive the first 72 hours of an initial emergency hospital visit’ (Malley, Malley, and Pajevic 2011, 5). If the same curve – the logistic curve – can describe quite different situations (the growth of a population, the probability that someone will die), then we can begin to see that sigmoid functions, and the logistic curve in particular, might be useful devices as approximations to the ’underlying function that generated the data.

The cost of curves in machine learning

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

How does this take place practically? As I have already mentioned, the logistic function appears frequently in machine learning literature, prominently as part of perhaps the most classical learning machine, the logistic regression model, but also as a component in other techniques such as neural networks (see table 4.2 for a sample of well-cited publications). Descriptions of logistic regression models appear in nearly all machine learning tutorials, textbooks and training courses (see Chapter 4 in (Hastie, Tibshirani, and Friedman 2009)). Logistic regression models are heavily used in biomedical research, where, as ‘logistic regression is the default “simple” model for predicting a subject’s group status’ (Malley, Malley, and Pajevic 2011, 43). As Malley et.al. suggest, ‘it can be applied after a more complex learning machine has done the heavy lifting of identifying an important set of predictors given a very large list of candidate predictors’ (43). Especially in comparison

to more complicated models, logistic regression models are relatively easy to interpret because they are superimposed on the linear model that we have been discussing already (see figure 4.2a and also chapters 2 and 3). As Hastie et.al write: ‘the logistic regression model arises from the desire to model the posterior probabilities of the K classes via linear functions in x , while at the same time ensuring that they sum to one and remain in $[0, 1]$ ’ (Hastie, Tibshirani, and Friedman 2009, 119). Paraphrased somewhat loosely, this says that the logistic regression model predicts what class or category a particular instance is likely to belong to, but ‘via linear functions in x .’ We see something of this predictive desire from the basic mathematical expression for logistic regression in a situation where there are binary responses or $K = 2$:

$$Pr(G = K|X = x) = \frac{1}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \beta_l^T x)} \quad (4.2)$$

(119)

In equation 4.2, the logistic function operates on linear functions and thus encapsulates lines in curves. That is, the linear model (the model that fits a line to a scattering of points in vector space) appears as $\beta_l 0 + \beta_l^T x$ (where as usual β refers to the parameters of the model and x to the matrix of input values). The linear model has, however, now been put inside the sigmoid function so that its output values no longer increase and decrease linearly. Instead they follow the sigmoid curve of the logistic function, and range between a minimum of 0 and a maximum of 1. As usual, small typographic conventions express some of this shift. In equation 4.2, some new characters appears: G and K . Previously, the response variable, the variable the model is trying to predict, appeared as Y . Y refers to a continuous value whereas G refers to membership of a category or class (e.g. survival vs. death; male

vs female; etc.).

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

10. I leave aside any further discussion of probability in machine learning here. It is the topic of chapter 5.

The cost of curves in machine learning

Whether or not the logistic function is a useful approximation to ‘the function that underlies the predictive relationship between input and out,’ it combines the features of the common vector space with a reference to a categorical state of affairs in the world. (Other techniques – neural networks or support vector machines – construct different kinds of reference.) The way in which we have ‘learned’ the logistic function by taking a textbook formula expression of it, and plotting the function associated with it is not the way that machine learners typically ‘learns’ an approximation to the predictive relationship between the input data and the output variables (the so-called ‘response variable’). Finding a function in practice means optimising parameters on the basis of the data. This is not a matter of mathematical analysis, but of algorithmic repetition.¹¹

If we turn just to the diagrammatic forms associated with logistic regression in *Elements of Statistical Learning*, something quite different and much more complicated than calculating the values of a known function is going on.

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

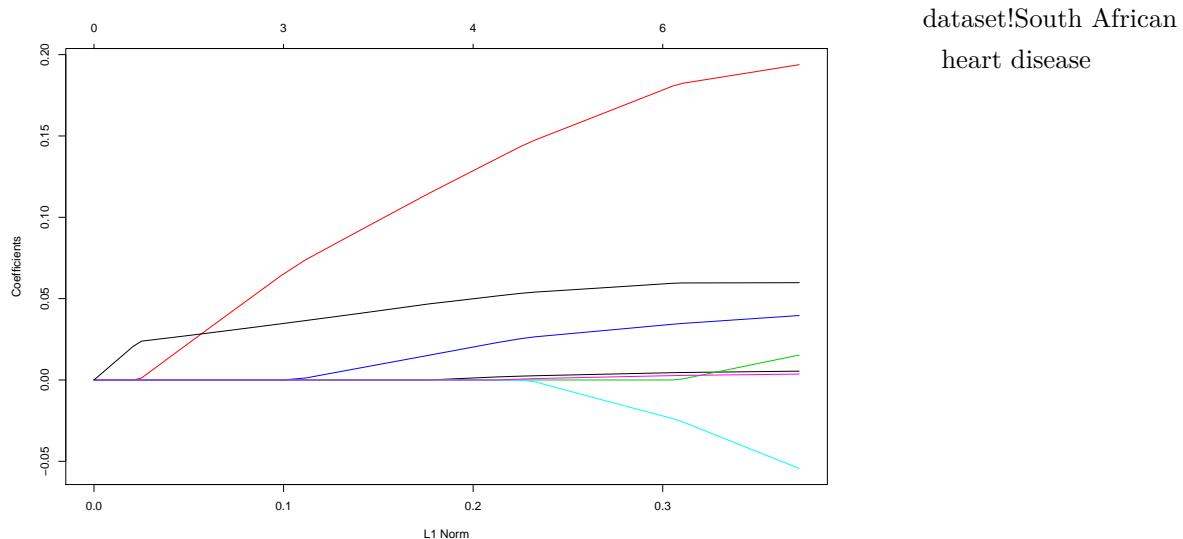


Figure 4.4: South African Heart disease regularization plot

For instance, in their analysis of the South African coronary heart disease data, Hastie and co-authors repeatedly model the risk of occurrence of heart disease using logistic regression. They first apply logistic regression fitted by ‘maximum likelihood’, and then by ‘L1 regularized logistic regression’ (126). The results of this function-finding work appear as tables of coefficients or as ‘regularization plots.’ As is often the case in *Elements of Statistical Learning*, it is assumed that readers already understand conventional statistical usages of logistic regression. Discussion dwells instead on different ways of fitting models or finding the values of parameters that control the operation of the function.

Figure 4.4 shows a series of lines. Each line shows the changing importance of a particular variable – obesity, alcohol consumption, weight, age, designated by numbers shown on the right hand side – as it is included in the logistic regression model in a different way. The use of the logistic function does not move through data radically differently. The same sigmoid curve has been used to model binary responses for more than half a century. The learning or function finding diagrammed in figure 4.4 concerns variations

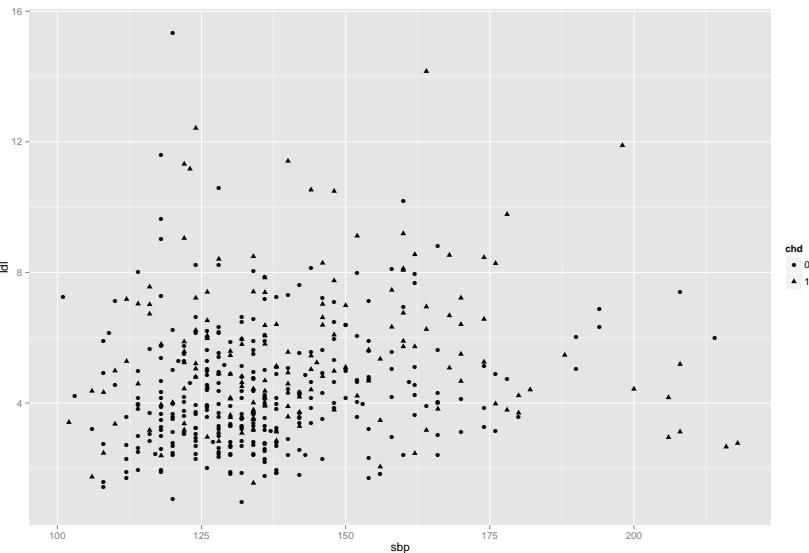


Figure 4.5: South African Heart disease decision plane

in parameters and ways of automating the variation of parameters beyond that undertaken by modelling experts such as statisticians and scientists when they fit models to data.¹²

The diagram of the changing parameters in a logistic regression model (Figure 4.4 suggests a constrained yet dynamic process of function finding. Machine learners re-shape the parameters of the function within a set of constraints or limits. We saw that the classic statistical model of linear regression fits lines to the data through the method of ordinary least squares (see chapter 3, equation 3.3). The combination of all the variables can be imagined as a surface whose contours include peaks and valleys in common

12. As we saw in chapter 3, the production of new tables of numbers that list the parameters of models that superimpose different dimensions of the data is a core operation of machine learning. Many of the plots and tables found in machine learning texts, practice and code offer nothing else but measurements of how the model parameters weight different dimensions of the vector space. In the case of logistic regression, the shape of the curve is determined using maximum likelihood. For present purposes, the statistical significance of this procedure is less important than the algorithmic implementation. This is the opposite to what might appear in a typical statistics textbook where the implementation of maximum likelihood would normally be quickly passed over. For instance, in *An Introduction to Statistical Learning with R*, a textbook focused on using R to implement machine learning techniques, the authors write: ‘we do not need to concern ourselves with the details of the maximum likelihood fitting procedure’ (James et al. 2013, 133).

vector space. Many different planes more or less closely fit the contours, and there may be no obvious best one. As we have seen, a linear model tries to find a line or plane or hyperplane (a higher dimensional plane) that aligns with this topography as closely as possible. In mathematical practice more generally, problems are posed and often solved in terms of finding functions. Mathematics textbooks are replete with demonstrations of this problem-solving activity, and mathematics is in large part practiced in solving problems by finding either values or functions that solve problems. These solutions are closed form or analytical solutions.¹³

Several obstacles hinder the construction of models using closed form approximate solutions. Unique ‘closed form’ or analytical solutions are quite unusual in machine learning. While they do exist for linear regression, they don’t exist for logistic regression nor for complex machine learners. Equally problematically, the closed form solution is run once, and the model it produces is subject to no further variation. The parameters define the line of best fit. It can be interpreted by the modeller in terms of p or R^2 or other measures of the model’s fit. But the model itself does not generate variations.

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

machine

learner!optimisation of

function!as partial
observer

Cost, loss, objective and optimisation

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

Optimisation is a practice of reference. ‘Science brings to light partial observers in relation to functions within systems of reference’ write Gilles Deleuze and Félix Guattari in their account of scientific functions (Deleuze and Guattari 1994, 129).¹⁴ In many machine learning techniques, for instance, the search for an optimised approximation to the function that generated the data is guided by another function called the ‘cost function’ (also known as the ‘objective function’ or the ‘loss function’; all the terms are somewhat evocative). Typically, machine learning problems are framed

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

in terms of minimizing or maximizing the cost function. The value of the cost function is usually understood in terms of errors, and minimizing the cost function implies minimizing the number of errors made by a machine learner. As we saw earlier, in his formulation of the ‘learning problem’, the learning theorist Vladimir Vapnik speaks of choosing a function that approximates to the data, yet minimises the ‘probability of error’ (Vapnik 1999, 31). Even the apparently simplest data modelling procedure of fitting a line to a set of points is usually implemented as an optimisation process in machine learning settings.

The cost function measures how predictions generated by a machine learner compare to known values in the data set. Every cost function implies some measure of the difference or distance between the prediction and the values actually measured. In classifying outcomes into two classes (the patient survives versus patient dies; the user clicks versus user doesn’t click; etc.), the cost function has to express their either/or outcome. Crucially, if cost functions re-configure ‘the act of fitting a model to data as an optimization problem’ (Conway and White 2012, 183), function finding occurs iteratively not analytically. Given a cost function, a machine learner can shape its variation of models parameters keeping in view – or partially observing – whether the cost function increases or decreases. The cost functions add a diagrammatic layer in which the relation between models, and particularly their predictive reference becomes visible. If there is learning here, it is not some enigmatic form or a higher form of scientific reason. Just the opposite, the cost function does something more like create a place from which variations in models can be viewed. A typical and widely-used cost function associated with logistic regression is defined as:

$$J(\beta) = \sum_{i=1}^m y_i \log h(x_i) + (1 - y_i) \log(1 - h(x_i)) \quad (4.3)$$

where

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

Equation 4.3 enfolds several manipulations and conceptual framings (particularly the Principle of Maximum Likelihood, a statistical principle; see chapter 5). But key terms stand out. First, the cost function $J(\beta)$ is a function of all the parameters (β) of the model. Second, the function defines a goal of maximising the overall value of the expression. The *min* describes the results of the repeated application of the function. Third, the heart of the function is a kind of average: it adds (Σ) all the values where the probability of the predicted class of a particular case $h(x_i)$ matches the actual class, and subtracts ($1 - y$) all the cases where the probability of the predicted class does not match the actual class. This so-called *log likelihood* function can be maximised, but not solved in closed form. The optimal values for β , the model parameters that define the model function need to be found through some kind of search.

Gradients as partial observers

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

Repeat until convergence:

$$\beta_j := \beta_j + \alpha(y_i - h_\beta(x_i))x_{\beta j} \quad (4.4)$$

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

Given that this expression encapsulates the heart of a major optimisation technique, we might first of all be struck by its operational brevity. This is not an elaborate or convoluted algorithm. As Malley, Mally and Pajevic observe, ‘most of the [machine learning] procedures ... are (often) nearly trivial to implement’ (Malley, Malley, and Pajevic 2011, 6). Note that this expression of the algorithm, taken from the class notes for [Lecture 3] of Andrew Ng’s ‘Machine Learning’ CS229 course at Stanford (see Figure 4.6), mixes an algorithmic set of operations with function notation. We see this in several respects: the formulation includes the imperative ‘repeat until

$$\begin{aligned}
 L(\theta) &= P(y|X;\theta) = \prod_i p(y^{(i)}|X^{(i)};\theta) \\
 &= \prod_i y^{(i)} h_\theta(x^{(i)})^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{1-y^{(i)}}
 \end{aligned}$$

Find θ that $\max L(\theta)$;

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

Apply gradient descent algorithm \rightarrow gradient ascent

$\theta := \theta + \alpha \nabla_{\theta} L(\theta)$: max the quadratic

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

$$\frac{\partial}{\partial \theta_j} L(\theta) = \sum_{i=1}^m (y^{(i)} - h_\theta(x^{(i)})) x_j^{(i)} \quad \begin{matrix} \text{dots of} \\ \text{algebra} \end{matrix}$$

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

\approx (batch gradient descent)

Figure 4.6: Gradient ascent for logistic regression

convergence'; it also uses the so-called 'assignment operator' $:=$ rather than the equality operator $=$. The latter specifies that two values or expressions are equal, whereas the former specifies that the values on the right hand side of the expression should be assigned to the left. Both algorithmic forms – repeat until convergence, and assign/update values – owe more to techniques of computation than to mathematical abstraction. In this respect more generally, by looking at implementations we begin to see how systems of references are put together. In this case, the specification for the gradient descent algorithm starts to bring us to the scene where data is actually reshaped according by the more abstract mathematical functions we have been describing (mainly using the example of the linear regression and its classic algebraic form $y = \beta_0 + \beta_1 x_1 + \dots + \beta_n x_n$). At the heart of this reshaping lies a different mathematical formalism: the partial derivative, $\frac{\partial}{\partial \beta_j} J(\beta_j)$. Like all derivatives in calculus, this expression can be interpreted as a rate; that is as the rate at which the cost function $J(\beta)$ changes with respect to

the different values of β ¹⁵. If there is any learning in machine learning, it pivots on the observation of movement. The partial derivatives in the gradient descent algorithm observe the direction in which the cost function becomes smaller. On each iteration of the algorithm, the parameters β of the model move in that direction of reduced cost, and perhaps less error.

function!diagrammatic
operation of
diagram!mathematical
function as
diagram|diagonal

The power of machine learning to learn, then, pivots around functions in disparate ways: the diagramming of functions in mathematical expressions, in the graphic forms of plots of lines and curves of the changing shape of functions, and in the algorithms that superimpose new functions – cost, loss or objective functions – in an iterative process of observation and modification. Machine learning diagrammatically distributes learning between people and things. People looking at curves, functions compressing data into parameters that support classification or predictions, and algorithms observing gradients. In several senses, people and machines together move along curves. The logistic function folds the lines that best fit the data into a probability distribution that can be read in terms of classification. The loss functions seek convergence between the predicted values and the known values found in the common vector space. The gradient-based algorithms look for the direction in which the peaks or valleys lie. Everywhere the diagram switches and diagonalizes between the human observers and machine observers. Every observer in this domain is partial, since the humans cannot see lines or curves in the multi-dimensional data, the functions that underpin models such as logistic regression or linear regression can traverse data in the common vector space, but can't show how well they see it, and the processes of optimisation only see the results of the model and its errors, not anything in its referential functioning. Impartiality, whether fully supervised or completely unsupervised, is impossible here.

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

function!as partial
observer
diagram!of power
Deleuze, Gilles!see also diagram!of
power

Amidst this chronic partiality, we can begin to understand the multiplication of functions and machine learners. Machine learners are functions that classify, predict and rank, but the function that defines a ‘machine learner’ contracts a range of partial observers opaquely associated with each other.

If, as I have been suggesting, the algorithmic power of machine learning derives from functions (and I think in a literal technical sense, that is what much machine learning literature states at length in great detail), its operational and referential power depends on the diagrammatic and sometimes experimental relations between different practices of observing. Attending to specific mathematical functions in isolation – the logistic function, the Lagrangean, the Gaussian, the quadratic discriminant, etc. – will not tell us how the operational power of functions comes together in machine learning, but it may provide ways of mapping the diagrammatic connections that generate reference operational and referential power. In his account of diagrams of power, Gilles Deleuze writes:

every diagram is intersocial and constantly evolving. It never functions in order to represent a persisting world but produces a new kind of reality, a new model of truth. ... It makes history by unmaking preceding realities and significations, constituting hundreds of points of emergence or creativity, unexpected conjunctions or improbable continuums (Deleuze 1988, 35)

This somewhat theoretical description of diagrams might help make sense of the learning in machine learning. Functions in machine learning are ‘intersocial’ in the sense that they bring together very different mathematical, algorithmic, operational and referential processes. The sigmoid function switches the geometry of the linear model over into the calculation of probabilities and classification. The cost functions re-craft statistical modelling as a quasi-iterative process of model generation and comparison. New kinds

of realities arise in which the classifications and predictions generated by the diagonal connections between mathematical functions and operational processes of optimisation can constitute a ‘new model truth’ and can unmake ‘preceding realities and significations.’ And despite my deliberately narrow focus on a single set of diagonals that connects linear models, the logistic function, the cost function and gradient ascent, there are hundreds and perhaps and hundreds of thousands of ‘points of emergence’ associated with this diagram of functioning. They are scattered across sciences, industry, government and commerce, and appear in manifold variations.

Lazzarato,
Maurizio!asemiotic
machine

Agency dances along the curves. Curves traverse and encapsulate many lines. They become mobile partial observers of many lines. Animating the curves, and then looking at those optimising animations as ‘learning’ generates operational power dynamics. The diagram attracts infrastructural, technical, professional, semiotic and financial diagonals that render its traits more real, more thickly transformative and, it goes almost without saying, more performant. In the history of automata, automation and animation, kinetic lures have long exercised fascination, and this may be part of the effect of machine learning. Yet such performant diagrams generate referential effects. Machine learning becomes ontologically potent. As Maurizio Lazzarato writes in *Signs and Machines*, ‘ontological mutations are always machinic. They are never the simple result of the actions or choices of the “man” who, leaving the assemblage, removes himself from the non-human, technical, or incorporeal elements that constitute him – all that is pure abstraction’ (Lazzarato 2014, 83). The differences of function held together in machine learners (bearing in mind the double meaning of that term) hold together diagrammatically, but generate statements and visual objects that count as knowledge, truth and increasingly as action.

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

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

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

Year	Title	Keywords	Cita- tions
1997	Comparing Support Vector Machines With Gaussian Kernels To Radial Basis Function Classifiers	clustering; pattern recognition; proto-types; radial basis function networks; support vector machines	392
1999	Three Machine Learning Techniques For Automatic Determination Of Rules To Control Locomotion	adaptive-network-based fuzzy inference system (anfis); artificial neural networks (ann's); functional electrical stimulation (fes); inductive learning (il); multilayer perceptron; radial basis function (rbf) ann; walking	41
2000	Constructing Fuzzy Models With Linguistic Integrity From Numerical Data Afreli Algorithm	data mining; function approximation; fuzzy modeling; knowledge extraction	80
2001	Greedy Function Approximation: A Gradient Boosting Machine	function estimation; boosting; decision trees; robust nonparametric regression	900
2001	Shared Kernel Models For Class Conditional Density Estimation	classification; density estimation; expectation-maximization (em) algorithm; mixture models; probabilistic neural networks; radial	29

Year	Title	Keywords	Cita- tions
2010	Regularization Paths For Generalized Linear Models Via Coordinate Descent	Lasso; Elastic Net; Logistic Regression; L(1) Penalty; Regularization Path; Coordinate Descent	912
2010	Regularization Paths For Generalized Linear Models Via Coordinate Descent	Lasso; Elastic Net; Logistic Regression; L(1) Penalty; Regularization Path; Coordinate Descent	859
2002	Evaluating Resource Selection Functions	Habitat Selection; Logistic Regression; Model Selection; Prediction; Rsf; Resource Selection Functions; Validation	587
2010	Regularization Paths For Generalized Linear Models Via Coordinate Descent	Lasso; Elastic Net; Logistic Regression; L(1) Penalty; Regularization Path; Coordinate Descent	565
2007	Random Forests For Classification In Ecology	Additive Logistic Regression; Classification Trees; Lda; Logistic Regression; Machine Learning; Partial Dependence Plots; Random Forests; Species Distribution Models	563
1995	Bayesian Computation And Stochastic Systems	Agricultural Field Experiments; Bayesian	495

ontology!stochastic
function!probability
distribution!Gaussian

Chapter 5

N = all: Probabilisation

In the final pages of *The Taming of Chance*, the philosopher Ian Hacking describes the work of the geodesic philosopher C.S. Peirce in terms of a twin affirmation of chance. On the one hand, Peirce, following the work of the psychophysicist Gustav Fechner and before him the astronomer-sociologist Adolphe Quetelet, makes the normal curve into an underlying reality.¹ The ‘personal equation,’ the variation in measurements made by any observer, becomes ‘a reality underneath the phenomena of consciousness’ (Hacking 1990, 205). At the same time, and in order to show the underlying reality of the normal curve, ‘Peirce deliberately used the properties of chance devices to introduce a new level of control into his experimentation. Control not by getting rid of chance fluctuations, but by adding some more’ (205). Peirce’s belief in absolute chance or a stochastic ontology, ‘a universe of chance’ as Hacking puts it, succeeded a series of ‘realizations’ of curves, in which first social, then biological and finally psychological variations were all understood as evidence of a generative function, the normal distribution or Gaussian function. In the century or so since, what happened to the thorough-going affirmation of statistical thought and probabilistic practice epitomised

1. The historian of statistics Stephen Stigler provides a lengthy account of Fechner’s work in (Stigler 1986, 239-259).

Hacking, Ian!on C.S.
Peirce

by Peirce? Hacking stresses that he does not understand Peirce as the precursor or the innovator of twentieth century statistical thought (Hacking's *Taming of Chance* ends at 1900), but rather as 'the first philosopher to conceptually internalize the way chance had been tamed in the nineteenth century' (215). What would the equivalent philosopher-machine learner internalize today? What would such persons, working in science or media or government, hold firm in relation to chance, probability and statistics?

In a broad sense, the setting that concerned Peirce in his work at the U.S. Government Coast Survey in the 1870s does not differ greatly from the setting that machine learners encounter today. In the opening lines of the First Edition of *Elements of Statistical Learning*, Hastie, Tibshirani and Friedman write:

The field of Statistics is constantly challenged by the problems that science and industry brings to its door. In the early days, these problems often came from agricultural and industrial experiments and were relatively small in scope (Hastie, Tibshirani, and Friedman 2009, xi)

At the end of the preface, they also cite, we might note in passing, Hacking's work: 'The quiet statisticians have changed our world' (xii). One of the challenges science and industry has brought to its door in recent years has been more data but also learning machines. With some justification, we might ask therefore: what difference do the 'vast amounts of data ... generated in many fields' (xi) make to the field of statistics? Statistics has, I will suggest in this chapter, gradually endowed machine learners with an increasing probabilistic semantics. This deeply affects how machine learners internalize their world. Conversely, we should ask: what do machine learners do to statistics? Is machine learning a further taming of chance? What role

does randomness and probability play in machine learning? These questions of how worlds becomes thinkable through machine learning can be addressed partly by contrasting the ‘taming of chance’ achieved by eighteenth and nineteenth century statistics and the statistical practices of machine learning today.

The broadest claim associated with statistical machine learning might be the simple expression shown in:

$$N = \forall X \quad (5.1)$$

In Equation 5.1, N refers to the number of observations (and hence the size of the dataset), the symbol \forall means ‘all’ since this is the level of inclusion which many fields of knowledge in science, government, media, commerce and industry envisage, and X refers to the data itself arrayed in common vector space. Note that this expression leaves some things out. Y , the response variable, for instance, may or may not be known. While both the expansion of data in the common vector space and the machine learners that traverse it have appeared in previous chapters, I focus here on changes in probability practices associated with machine learning, and in particular, $N = \forall X$, the claim that with all the data, statistical thinking and the potentials of the statistical reasoning fundamentally change. The claim that with $N = \forall X$ everything changes has been widely discussed.²

Viktor Mayer-Schönberger and Kenneth Cukier’s *Big Data: A Revolution That Will Transform How We Live, Work and Think* present this shift in many different ways in the course of the vignettes and comparisons that

2. Rob Kitchin provides a very useful overview of these claims in (Kitchin 2014). While I will not analyse the claims about ‘big data’ in specific cases in any great detail, the growing literature on this topic suggests that machine learning in its various operations – epistemic construction of common vector space, function finding as association of partial observers and a re-internalisation of probability – generates considerable difficulties and challenges for knowledge, power and production.

Mayer-Schönberger,
Viktor
Cukier, Kenneth

have become typical of the data revolution genre. In a chapter entitled ‘More,’ they sketch the transition from data practices reliant on sampling to data practices that deal with all the data:

Using all the data makes it possible to spot connections and details that are otherwise cloaked in the vastness of the information. For instance, the detection of credit card fraud works by looking for anomalies, and the best way to find them is to crunch all the data rather than a sample (Mayer-Schönberger and Cukier 2013, 2013, 27)

In the several hundred pages that follow in *Big Data*, the problem of how to ‘crunch all the data’ is not a major topic. While they mention the role of social network theory (30), ‘sophisticated computational analysis’ (55), ‘predictive analytics’ (58) and ‘correlations’ (7), and they observe that ‘the revolution’ is ‘about applying math to huge quantities of data in order to infer probabilities’ (12), any further consideration of a change in statistical practices subsumed in ‘infer probabilities’ is largely confined to a business-oriented contrast between having some of the data and having all the data (that is, businesses often have all the data on their customers). This is not to criticize a book that sets out to describe trends affecting business and government for a general readership, but without a sense of how statistical thinking animates almost all salient features of crunching the data and particularly the predictions, it becomes hard to see how the ‘revolution’ takes place. Just as nineteenth century statistics transformed measurements (for instance, the mean as average of all measured values) into real quantities (for instance, mean as the ideal or abstract property of a population; e.g. life expectancy), the shift between n and $\forall X$, a shift very much animating and dependent on machine learning, internalizes, I will suggest, probability and chance into machinic operations. This is

a statistical event akin to the advent of the Normal distribution (and \ indeed, N is a standard symbol for the Normal distribution in statistics textbooks) as a way of knowing and controlling populations (Hacking 1975, 108). To signal its continuity with the invention of probability, I term it here ‘probabilisation,’ a pleonasm that refers to rendering in terms of probabilities. probabilisation|seealso{diagram!probabilisation of}

statistics!history of!from error to real quantity statistics!probability distributions!normal

Machine learning as statistics inside out

In *The Taming of Chance*, Hacking argues that modern statistical thought transposed a way of calculating errors in experimental measurements and astronomical observations into real quantities typically described by the normal distribution. This transposition or inversion relied on four intermediate steps passing through probability calculus (particularly the work of Jacob Bernoulli and the binomial or heads-tails probability distribution in the 1690s (143)), on large numbers of measurements (the most famous being the chest measurements of soldiers in Scottish regiments, but these were only one flurry amidst an avalanche of numbers in the 1830-1840s), on an idea of multiple, minute independent causes producing events (particularly as developed in medicine but also in studies of crime), and the ‘law of errors’ applying to measurements made by, amongst others, astronomers (Hacking 1990, 111-112). As Hacking observes, coins, suicides, crime, chest measurements, and astronomical observations all accumulate in a picture of statistical reality which remains, although somewhat altered, indelible in contemporary statistical thought in its frequent recourse to probability distributions. In this entanglement, observers and the observed changed places. The distribution of errors made by astronomers measuring the position of stars or planets became a distribution or variation inherent in things. All of this seems a long way from machine learning, and in terms

of years, the work of figures such as Poisson, Laplace, Quetelet and even Galton, is well-removed. In terms of tables and functions (the concerns of the preceding two chapters), the distance is not so great. There is greater variety in tables (partly due to the common vector space) and in functions, but the entwining or even swapping between what relates to an observation and what concerns the real, continues. Machine learners engage in that swapping or re-distributing of numbers all the time. Viewed from the standpoint of Hacking, machine learning reverse-engineers the invention of modern statistical thinking. It takes back the ‘real quantities’ that modern statistics had attributed to the populations of the world and puts them into devices, machine learners that people then observe, monitor and indeed measure again in many ways. The direct swapping between uncertainty in measurement and variation in real attributes that statistics achieved now finds itself re-routed and intensified as machine learners measure the errors, the bias and the variance of devices.

This swapping or re-distribution is not a simple mirror-image reversal, as if machine learners mistake a device for the world. Machine learners do constantly take statistical thinking as a basic semantic frame for their devices. When Hastie and co-authors write (as we saw in the previous chapter) ‘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), they invoke the ‘real quantities’ first elaborated and articulated by proto-statisticians such as Quetelet grappling with population and sample parameters. The major structuring differences in machine learning as a field of knowledge-practice show the marks of increasingly strong commitment to the reality of the statistical.

Reading and working with machine learning techniques usually means encountering and responding to some of that statistical apparatus drawn from

parametric	non-parametric	
bias	variance	machine
prediction	inference	learning!structure
generative	discriminative	differences

Table 5.1: Some structuring differences in machine learning

statistics, but these are not typically the statistical tests of significance or variation. In contrast to a statistics textbook such as the widely used *Basic Practice of Statistics* (Moore 2009) or even a more advanced guide such as *All of Statistics* (Wasserman 2003), where statistics (t-test, chi-squared test, etc.) hypothesis testing, and analysis of uncertainties (confidence intervals, etc) order the exposition, the machine learning texts describe a conceptual apparatus curiously unadorned by the plethora of methods found in mainstream statistics. Statistical underpinnings may be fundamental, but this does not mean that machine learners simply automate statistics.³ Instead, a basic set of contrasts or indeed oppositions that owe much to probabilistic thinking pattern statements on machine learning. The contrasts shown in Table 5.1 all have a statistical facet and anchoring to them. Some refer to errors that affect how a machine learner refers to data (bias and variance; see discussion below); some designate an underlying statistical intuition about how particular machine learners treat data (does the model seek to generate the data or classify – discriminate – it; e.g. Naive Bayes or Latent Dirichlet Allocation are *generative* models whereas logistic regression or support vector machines are *discriminative*); parametric and non-parametric

3. Leo Breiman writing in 2001 during the heyday of academic development of machine learning argues, describes the ‘two cultures’ of statistics: ‘in the past fifteen years, the growth in algorithmic modeling applications and methodology has been rapid. It has occurred largely outside statistics in a new community—often called machine learning—that is mostly young computer scientists (Section 7). The advances, particularly over the last five years, have been startling’ (Leo Breiman 2001, 200). Soon afterwards, statisticians such as Hastie, Tibshirani and Friedman publish the first major statistical machine learning textbooks (Hastie, Tibshirani, and Friedman 2001).

machine learning!structuring differences

diagram!reference machine learners!ontological mutations of

machine learning!probabilisation of|sealsoprobabilisation

function!probability distributions

describe the role of probability distributions in the model; and others indicate different kinds of statistical knowledge practice (prediction seeks to anticipate while inference seeks to interpret, etc.; also see discussion below). These broad structuring differences reach down deeply into the architecture, the diagrams, the practices, statements and visual objects and computer code associated with $N = \forall X$. Because they anchor basic operations of machine learning in probability statements, techniques and formalisms derived from statistics have in the last two decades increasingly populated the field, furnishing and rearranging its diagrammatic references to the worlds of industry, agriculture, earth science, genomics, etc., but also, crucially, triggering ontological mutations in machine learners themselves.⁴ These structuring differences in machine learning arise from probabilisation.

While these structuring differences are practically very important, and deeply shape certain kinds of practice in machine learning, the underlying operator that allows swapping between knowledge and the world, and then between events and devices is probability thinking, and in particular, the functions that describe variations in probability, probability distributions. Historically, two probability distributions loom large. The binomial distribution, which describes the probability of the number of n ‘successes’ out of N trials or observations, was explored extensively in the seventeenth and eighteenth centuries in the context of games of chance (Hacking 1975, 57-134). Because it accommodates only two discrete outcomes, the binomial distribution anchors many machine learning classifiers. The normal distribution pervades nineteenth century statistical thinking as it generalizes across law, medicine,

4. Maurizio Lazzarato describes ontological mutations as ‘always machinic. They are never the simple result of the actions or choices of the “man” who, leaving the assemblage, removes himself from the non-human, technical, or incorporeal elements that constitute him – all that is pure abstraction’ (Lazzarato 2014, 83). Lazzarato’s account of diagrams as processes that slip past signification and representation echoes Deleuze and Guattari’s, but usefully highlights the plurality of machines that inhabit science, music, art, cities and markets.

agriculture, finance and not least, sociology. These distributions appear in countless variations in scientific, government and popular literature of many different kinds.⁵

random variable|seealso-
function!probability
distribution

$$f(x; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \quad (5.2)$$

The function shown in equation (5.2) is the so-called normal or Gaussian distribution. Its mathematics were intensively worked over during the late eighteenth and early nineteenth centuries in what has been termed ‘one of the major success stories in the history of science’ (Stigler 1986, 158), and it has a power-laden biopolitical history closely tied with knowledges and governing of national and other populations in terms of morality, health, and wealth (see (Hacking 1975, 113-124)). The key symbols here include μ , the mean and σ , the variance, a number that describes how widely dispersed the values of the variable, x are. These two parameters together describe the shape of the curve. Given knowledge of μ and σ , the normal or Gaussian probability distribution maps all outcomes to probabilities (or numbers in the range 0 to 1). Put more statistically, viewed in terms of functions such as the Gaussian distribution, events become random variables. Every variable potentially becomes a function: ‘a random variable is a mapping that assigns a real number to each outcome’ (Wasserman 2003, 19).

The possibility of treating all variables as random variables, that is, as probability distributions, was a significant historical achievement, but one that continues to develop and ramify.⁶ The concept of the random variable tends to lend ontological weight to probability. When conceptualised as real

5. Statistical graphics have a rich history and semiology that I do not discuss here (see (Bertin 1983)).

6. The mapping that assigns numbers to outcomes (heads v. tails; cancer v. benign; spam v. not-spam) is a probability distribution. As I have argued in (Mackenzie 2015b), random variables have become much more widespread in statistical practice due to changes in computational techniques.

\ quantities in the world rather than epiphenomenal by-products of inaccuracies in our observations or measuring devices, probability distributions weave directly into the productive operations of power. Distribution in the sense of locating, positioning, partitioning, sectioning, serialising or queuing operations pervades the development of disciplinary power, as Michel Foucault richly details (Foucault 1977), but in almost every setting, distribution in the sense of counting and weighting of different outcomes also operates. This constant interweaving of spatial, architectural, logistical and calculative processes has energised statistical thought for several centuries.⁷ distribution!see{function!probability distribution} For instance, given the normal distribution, it is possible, under certain circumstances, to effectively subjectify someone on the spot. If an educational psychologist indicates to someone that their intelligence lies towards the left-hand side of the normal curve peak (and hence less than the population mean), they quickly assign them to a potentially institutionally and economically consequential trajectory. Since its inception in the social physics of Adolphe Quetelet as a way of referring to a property of populations, the normal curve has promised or threatened to support the re-shaping and control of populations (in terms of health, morality and wealth). Since then, probability distributions and their corresponding curves have multiplied.

Subsequent statisticians developed dozens of different probability distributions to map continuous and discrete variations to real numbers. Other probability distributions abound — normal (Gaussian), uniform, Cauchy exponential, gamma, beta, hypergeometric, binomial, Poisson, chi-squared,

7. ‘Distribution’ pervades Foucault’s account of power and knowledge from *The Order of Things* (Foucault 1992 [1966]) onwards. Foucault treats distributions in several different ways: as spatial or logistical techniques, as mathematical orderings of large numbers of people or things, and as a methodological and theoretical framing device. In *Discipline and Punish* (Foucault 1977), the spatial sense prevails, but in later works, the population or demographic sense of distribution takes precedence (Foucault 1998). Distribution certainly has theoretical primacy in his account of power: ‘relations of power-knowledge are not static forms of distribution, they are “matrices of transformations”’ (99).

Dirichlet, Boltzmann-Gibbs distributions, etc. (see (NIST 2012) for a gallery of distributions) — because outcomes occur in widely differing patterns. The queuing times at airport check-ins do not, for instance, easily fit a normal distribution. Statisticians model queues using a Poisson distribution, in which, unfortunately for travellers, distributes the number of events in a given time interval quite broadly. Similarly, it might be better to think of the probability of rain today in north-west England in terms of a Poisson distribution that models clouds in the Atlantic queuing to rain on the north-west coast of England. (Rather than addressing the question of whether it will rain or not, a Poisson-based model might address the question of how many times it will rain today.)

If functions such as equation (5.2) have persisted for so long as operational underpinnings, what happen to them in machine learning? The pages of a book such as *Elements of Statistical Learning* show many signs of an ongoing re-distribution of probability distributions. Superficially we could simply note their abundance. Hastie and co-authors diversely invoke probability distributions. They speak of ‘Gaussian mixtures,’ ‘bivariate Gaussian distributions,’ standard Gaussian, ‘Gaussian kernels,’ ‘Gaussian errors’, ‘Gaussian assumptions,’ ‘Gaussian errors,’ ‘Gaussian noise,’ ‘Gaussian radial basis function,’ ‘Gaussian variables,’ ‘Gaussian densities,’ ‘Gaussian process,’ and so forth. The term ‘normal’ appears in an even wider spectrum of similar guises, but they uniformly entail treating events, things, properties or attributes as probability distributions. The wide distribution of the Gaussian function does not mean that one function, the Gaussian probability density function reigns supreme above all machine learners.

The diverse curves of probability distributions — and we will see below some reasons why we can expect them to proliferate in certain settings — attests to the variety of ways in which events (occurrence of cancer, occurrence

machine learner!probability distribution as control surface

of the word ‘Viagra’ in an email, the click on a hyperlink, etc.) might be mapped to real numbers. Despite the sometimes dense mathematical diagrammatics, the term *distribution* emphasises a tangible and practically resonant way of thinking about how events or possible outcomes shift about as the parameters of a function vary. Machine learners adjust these parameters in different ways. For instance, parametric and non-parametric models (see table ??) differ in that the former have a limited number of parameters and the latter an undefined number of parameters (for instance, Naive Bayes, k nearest neighbours or support vector machine models). But both kinds assume that an underlying probability distribution – a function, ‘unobservable’ or not – operates, even if it changes with new data. A probability distribution under these assumptions becomes the closest reality we have to whatever process generated all the variations in data gathered through experiments and observations. From a probabilistic perspective, the task of machine learning is to estimate the parameters (the mean μ and variance σ in the case of Gaussian curve) that shape of the curve of the probability distribution. Given the shape of that curve, many inferences and predictions become possible. The probability distribution function is a crucial control surface for machine learning understood as a form of movement through data. In contrast to the gradual endowment of realities with probability that we see in the history of statistics (and later in natural sciences such as physics and biology), statistical machine learning endows devices and control systems with probability.

Naive Bayes and the distribution of probabilities

What does this mean in practice? The mathematical expression for one of the most popular of all machine learning classifiers, the Naive Bayes classifier, stands out for its simplicity and lack of parameters.

$$f_j(X) = \prod_{k=1}^p f_{jk}(X_k) \quad (5.3)$$

(Hastie, Tibshirani, and Friedman 2009, 211)

Some machine learners are so simple that they can be implemented in a few lines of code. Their simplicity, however, belies their power. The function shown above in equation (5.3) is about the simplest one to be found in most machine textbooks yet easily adapts for high dimensional data, the kind of data associated with $N = \forall X$.⁸ While the Naive Bayes classifier is one of the most popular machine learning algorithms, it is more than 50 years old (Hand and Yu 2001). The key diagrammatic elements of the classifier as expressed in the equation are \prod , an operator that multiplies all the values of the matrix of X values (from 1 to p) to generate a product. What product does the Naive Bayes classifier produce? The expression $f_j(X)$ refers to a probability density; that is, it describes the probability that a particular thing (a document, an image, an email message, a set of URLs, etc.) belongs to the class of things j . In constructing this estimate of the probability that a thing is an instance of class j , p different features of the thing are taken into account. (To use the language of chapter 3, the subscript k indexes the p dimensions of the common vector space.) The subscripts $k = 1$ on the \prod operator, and k on the data X_k indicate that the Naive Bayes classifier

8. The other contender for simplest machine learner would be the also very popular k nearest neighbours. As Hastie et. al. observe: ‘these classifiers are memory-based and require no model to be fit’ (Hastie, Tibshirani, and Friedman 2009, 463). Like the Naive Bayes classifier, the equation for k nearest neighbours is simple:

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i \quad (5.4)$$

where $N_k(x)$ is the neighborhood of x defined by the k closest points x_i in the training sample (14).

In equation 7.3, a parameter appears: k , the number of neighbours. This contrasts greatly with the linear models discussed in chapters 3 and 4 where the number of parameters p usually equals the number of variables in the dataset or dimensions in the common vector space.

probability!conditional

makes use of a series of features or variables in calculating the overall probability that a given thing or observation belongs to a specific class. The classifier produces a product $f_j(X)$ by calculating the *joint probability* of all the *conditional* probabilities of the features or predictor variables in X for the class j . As *Elements of Statistical Learning* rather tersely puts it, ‘each of the class densities are products of the marginal densities’ (Hastie, Tibshirani, and Friedman 2009, 108).

Almost everything about the Naive Bayes classifier concerns probability (including its name, with its reference to the Bayes Theorem, an important late eighteenth century concept), yet there is little obvious connection to statistics in its modern form of tests of significance. As Drew Conway and John Myles-White write in *Machine Learning for Hackers*,

At its core, [Naive Bayes] ... is a 20th century application of the 18th century concept of *conditional probability*. A conditional probability is the likelihood of observing some thing given some other thing we already know about (Conway and White 2012, 77)

They point here to the role of ‘conditional probability,’ a probability conditioned on the probability of something else. Conditional probability lies at the heart of many of the data transformation associated with prediction or pattern recognition since it links different variable or features together as we see in Naive Bayes often by simply multiplying probabilities.⁹ As any of the many accounts of the technique will explain, the name comes from Bayes Theorem, one of the most basic yet widely used results in probability

9. In (Mackenzie 2014a), I have suggested that the intensification of multiplication associated with probabilistic calculation may constitute an important mutation in the ontological and practical texture of numbers. The epidemiological modelling of H1N1 influenza in London 2009 involved multiplying a great variety of probability distributions in order to calculate the conditional probability of influenza over time.

theory (again dating from the eighteenth century), yet Naive Bayes does not even fully embrace Bayes Theorem as the principle of its operation. The classifier has a simple probabilistic architecture based on the concepts of conditional probability and joint probability; it calculates a probability density function $f_j(X)$ or probability distribution for each possible class of things. Its architecture is simple, however, because it makes a drastically reductionist assumption that features are independent of each other (hence the name ‘naive’), where ‘independent’ means that they do not affect each other, or that they have no relation to each other. We will see below that dramatic simplifications do not necessarily weaken the referential grasp of machine learners on the world, but in certain ways allow them to relate to it more directly.

Spam: when $\forall N$ is too much?

While equation (5.3) does not set out all the steps in transforming some data into a prediction of the class that a given instance, event, thing or observation belongs to, the code needed to do this are relatively brief. In *Doing Data Science*, Rachel Schutt and Cathy O’Neill furnish a bash script (that is, command line instructions) to download a well-known email dataset and build a Naive Bayes classifier that labels email as spam or not. In many ways, this is canonical machine learner pedagogy, and for Naive Bayes, email spam detection has become the standard example (Andrew Ng uses it in CSS 229, Lecture 5 ([Lecture 1 / Machine Learning \(Stanford\) 2008](#))). In this setting, machine learners operating as spam filters coping with too much communication. $\forall N$ can be a bother. Nevertheless, the concision of the small script, which fetches the Enron email dataset, calculates the ‘marginal densities’ or conditional probabilities for each word given how often it is associated with either spam or non-spam email, and then outputs

dataset!spam
email!Enron

the probability that a particular email is spam, is striking.¹⁰ These few dozen lines convert some part of a communicative reality – the Enron email dataset – into a predictive classifier that tests the world: is a given message spam or not?

A typical spam email in the Enron dataset, a dataset that derives from the U.S Federal Energy Regulatory Commission’s investigation into Enron Corporation (Klimt and Yang 2004), looks like this:

Subject: it's cheating, but it works ! can you guess how old
she is ? the woman in this photograph looks like a happy
teenager about to go to her high school prom, doesn't she ?
she's an international, professional model whose photographs
have appeared in hundreds of ads and articles whenever a client
needs a photo of an attractive, teenage girl.but guess what ?
this model is not a teenager ! no, she is old enough to have a
7-year-old daughter.. and...the model's real age is in her 30'
s.all she will say about her age to her close friends is, " i'm
dangerously close to 40." she also says, " if it weren't for this
amazing new cosmetic cream called 'deception,' i would lose
hundreds of modeling assignments...because...there is no way i
could pass myself off as a teenager." service dept 9420 reseda
blvd # 133 northridge, ca 91324

The text of a typical non-spam email like this:

10. The input to the script is a single word such as ‘finance’ or ‘deal’. The model is so simple that it only classifies a single word as spam. The `bash` script carries out four different transformations of the data in building the model. It uses only command line tools such as `wc` (word count), `bc` (basic calculator), `grep` (text search using pattern matching) and `echo` (display a line of text). These tools or utilities are readily available in almost any UNIX-based operating system (e.g. Linux, MacOS, etc). The point of using only these utilities is to illustrate the simplicity of the algorithmic implementation of the model. The first part of the code downloads the sample dataset of Enron emails (and I will discuss spam emails and their role in machine learning below). Note that this dataset has already been divided into two classes - ‘spam’ and ‘ham’ – and emails of each class have been placed in separate directories or folders as individual text files.

Subject: industrials suggestions..... -----forwarded
 by kenneth seaman / hou / ect on 01 / 04 / 2000 12 : 47
 pm----- - pat clynes @ enron 01 / 04 / 2000 12 :
 46 pm to : kenneth seaman / hou / ect @ ect, robert e lloyd /
 hou / ect @ ect cc : subject : industrials ken and robert, the
 industrials should be completely transitioned to robert as of
 january 1, 2000.please let me know if this is not complete and
 what else is left to transition . thanks, pat

These are both recognisable things in the world for anyone who uses email. How do they become X or even $f_j(X)$ in the Naive Bayes classifier? How do words in a document become probability distributions? The code is instructive:

```
#!/bin/bash

# file: enron_naive_bayes.sh
# description: trains a simple one-word naive bayes spam
# filter using enron email data
# usage: ./enron_naive_bayes.sh <word>
# requirements:
#     wget
#
# author: jake hofman (gmail: jhofman)
# how to use the code

if [ $# -eq 1 ]
then
    word=$1
else
    echo "usage: enron_naive_bayes.sh <word>"
    exit
fi
```

```

    fi

    ### PART 1

if ! [ -e enron1.tar.gz ]
then
    wget 'http://www.aueb.gr/users/ion/data/enron-spam/preprocessed/enron1.tar.gz'
fi

if ! [ -d enron1 ]
then
    tar zxvf enron1.tar.gz
fi

cd enron1

### PART 2

Nspam=`ls -l spam/*.txt | wc -l`
Nham=`ls -l ham/*.txt | wc -l`
Ntot=$Nspam+$Nham

echo $Nspam spam examples
echo $Nham ham examples

Nword_spam=`grep -il $word spam/*.txt | wc -l`
Nword_ham=`grep -il $word ham/*.txt | wc -l`
echo $Nword_spam "spam examples containing $word"
echo $Nword_ham "ham examples containing $word"

### PART 3

Pspam=`echo "scale=4; $Nspam / ($Nspam+$Nham)" | bc`
```

```

Pham=`echo "scale=4; 1-$Pspam" | bc`
echo
echo "estimated P(spam) =" $Pspam
echo "estimated P(ham) =" $Pham
Pword_spam=`echo "scale=4; $Nword_spam / $Nspam" | bc`
Pword_ham=`echo "scale=4; $Nword_ham / $Nham" | bc`
echo "estimated P($word|spam) =" $Pword_spam
echo "estimated P($word|ham) =" $Pword_ham

### PART 4

Pspam_word=`echo "scale=4; $Pword_spam*$Pspam" | bc`
Pham_word=`echo "scale=4; $Pword_ham*$Pham" | bc`
Pword=`echo "scale=4; $Pspam_word+$Pham_word" | bc`
Pspam_word=`echo "scale=4; $Pspam_word / $Pword" | bc`
echo
echo "P(spam|$word) =" $Pspam_word
cd ..

```

(Schutt and O’Neil 2013, 105-106)

After fetching the dataset from a website, the code counts the number of emails in each category and prints them, and then counts the number of times that the chosen word (e.g. ‘finance’ or ‘deal’) occurs in both the spam and non-spam or ham categories. Using these counts, the script estimates probabilities of any email being spam or ham, and then given that email is spam or ham, that the particular word occurs. To estimate a probability means, in this case, to divide the word count for the chosen word by the count of the number of spam emails, and ditto for the ham emails. In Part 4, the final transformation of the data, these probabilities are used to calculate the probability of any one email being spam given the presence of that word.

diagram!world

Again, the mathematical operations here are no more complicated than adding, multiplying and dividing. The probability that the chosen word is a spam word is, for instance, the probability of occurrence of the word in a spam email multiplied by the overall probability that an email is spam. Finally, given that the probability of the chosen word occurring in the email dataset is the probability of it occurring in spam plus the probability of it occurring in ham, the overall probability that an email in the Enron data is spam given the presence of that word can be calculated. It is the probability that the chosen word is a spam word divided by the probability of that word in general.

This slightly bamboozling and mechanical description of what the script does shows something of how the joint probability function in equation (5.3) diagrams the world. Not all machine learning models are so simple that they can be conveyed in 30 lines of code (including downloading the data and comments). This script indicates that nothing that occurs in relation to classification is intrinsically mysterious, elusive or indeed particularly abstract. On the contrary, the referentiality and operational power of classifiers takes place through the accumulated counting, adding, multiplying (that is, repeated adding) and dividing (that is, multiplying by parts or fractions) constrained by the probability distribution. Probability re-distributes things such as emails or documents as, in this case, events in a population of words. The Naive Bayes classifies endows every word found in the Enron datasets with a probability density function. The classification of each email becomes a matter of estimating a conditional probability based on the joint probability distribution that quantifies the chance of all the words in that email appearing together. Probabilities are always between 0 and 1, or between 0% and 100%, and classification entails selected a cutoff or dividing line. For instance, greater than 0.5 might

result in a classification as `spam`. In the `enron` dataset, ‘finance’ has a 0.69 probability!history of chance of being spam, while ‘sexy’ has a chance of 1. The re-distribution of textual forms – the emails – into probabilistic distributions is practically straightforward. Ironically, like the Naive Bayes classifier’s own reliance on seventeenth and eighteenth century probability calculus, the frequent application of this machine learner to document classification and retrieval echoes the seventeenth century thinking that first conceived of the very notion of ‘probability’ in relation to the evidential weight of documents (Hacking 1975, 85).

The improbable success of the Naive Bayes classifier

Naive Bayes classifiers have been surprisingly successful. Like statistics more generally, Naive Bayes treats the world as a set of probabilistic processes, as fields of random variables in process of change. The world comprises populations, and populations generate events. Emails have a real probability of being spam, and this probability is acted on many times every day for email users. There is something quite artificial at work in the construction of these populations and their associated probability distributions. They are intentionally artificial and limited. They do not correspond or refer directly to what we know, for instance, of how language works, but instead to a rather different set of concerns. They remain but less directly invested in a distribution as the generative reality of the population. Like most machine learning techniques encountering complex realities, classifiers such as Naive Bayes ignore many obvious structural features of emails as documents (for instance, word order, or co-occurrences of words). Yet this very artificiality or simplicity in their reference to the world allows machine learners to

appear in many different guises.

```
[1] 5358 11 [1] "index" "anchor" "DE" "AU" "SC" "WC" "TI"
```

```
[8] "PY" "TC" "UT" "CR"
```

```
## Error in order(nb2$PY): argument 1 is not a vector
```

```
## Error in print(tab, type = "latex", row.names = FALSE): object 'tab' no
```

The altered relation between modern statistical and machine learning practice starts to appear in Naive Bayes from the early 1990's as statisticians begins to generalize and re-diagram Naive Bayes by examining its statistical properties more carefully. Table ?? shows 30 of the most cited Naive Bayes-related scientific publications.¹¹ The list of titles sketches a double movement. On the one hand, we see the typical diagonal movement of a machine learner across disciplines – computer science, statistics, molecular biology (especially of cancer), software engineering, internet portal construction, sentiment classification, and image ‘keypoint’ recognition. On the other hand, highly cited papers such as (Friedman 1997) and (Hand and Yu 2001) point to an intensified statistical treatment of machine learners themselves taking place during these years, an intensified observation of machine learners that strongly affects their ongoing development (leading, for instance, to the much more heavily probabilistic topic models appearing in the following decade (Blei, Ng, and Jordan 2003)).

In *Elements of Statistical Learning*, Hastie, Tibshirani and Friedman characterise the capacity of the Naive Bayes classifier to deal with high dimensional data:

¹¹. Citation counts, even from the more reliable Reuters-Thomson Web of Science database, are difficult to evaluate when moving between disciplines. Some fields, such as computer science and biology, publish huge numbers of papers compared to smaller disciplines such as astronomy or plant ecology.

It is especially appropriate when the dimension p of the feature space is high, making density estimation unattractive. The naive Bayes model assumes that given a class $G = j$, the features X_k are independent (Hastie, Tibshirani, and Friedman 2009, 211).

probabilisation!
ancestral
communities of
machine
learner!
statistical
decomposition of

Similar formulations can be found in most of the machine learning books and instructional materials currently available. This appropriateness relates directly to $\forall N$, and the expansion of the common vector space of data. As we saw above in equation (5.3), p stands for the number of different dimensions or variables in the data set. In the spam classifier, the number of dimensions is probably quite large because every unique word adds a new dimension to the ‘feature space’ or common vector space. Compared to the complications of logistic regression, neural networks or support vector machines, 5.3 seems incredibly simple. How is it that a simple multiplication of probabilities and the assumption that ‘features ... are independent’ can, as Hastie and co-authors write: ‘often outperform far more sophisticated alternatives’ (211)?

The answer to this conundrum of success does not lie in statistics or in the increasingly availability of data to train machine learners on. I want to explore two other ways to view the improbable success of Naive Bayes. Both contrasts – inference/prediction, bias/variance – appeared in the table 5.1. Both are germane to the broader question of how machine learning re-iterates the ‘taming of chance’ accomplished by modern statistics. The first way to view this success is in terms of *ancestral communities* of probabilisation. The second concerns the statistical decomposition of machine learners in terms of their statistical bias and variance.

Ancestral probabilities in documents: inference and prediction

Along with linear regression, Naive Bayes might be *the* standard introductory machine learning technique. The Naive Bayes classifier is almost always demonstrated on the problem of filtering spam email (Conway and White 2012; Schutt and O’Neil 2013, 93-113; Kirk 2014, 53; Lantz 2013, 92-93; Flach 2012; *Lecture 6 / Machine Learning (Stanford) 2008*), and in particular dealing with the abundance of spam emails concerning a drug for erectile dysfunction sold under the tradename ‘Viagra’ (a drug that was itself the byproduct of the clinical trial for hypertension and heart disease). What are we to make of this reiteration of the same problem? Admittedly spam, and spam trying to sell Viagra in particular, has been a very familiar part of most email users since 1997 when Viagra was approved for sale, and of all the documents that machine learners mundanely encounter in quantity in those years, email might be the most numerous as well as one of the simplest. (The other would be scientific publications. Many more recent machine learners train as classifiers on scientific publications (Blei and Lafferty 2007)) Naive Bayes classifiers and variations of them also became practical devices in managing email traffic for most people, whether they know it or not, during the mid-1990s (see for instance, [SpamAssassin](#)).

The constant reiteration of email spam filtering using Naive Bayes is also a side-effect of another process, a process akin to the transposition that attributed probability distributions to populations in the nineteenth century. Like many machine learners, the genealogy of Naive Bayes has one important lineage derived from documents and the problem of their classification and retrieval. This genealogical affiliation with a particular problem such as document classification (or image recognition) generates many re-iterations

and versions of machine learners over time. As Lucy Suchman and Randall Trigg wrote in their study of work on artificial intelligence,

Suchman, Lucy; Trigg, Randall
ancestral communities

rather than beginning with documented instances of situated inference ... researchers begin with ... postulates and problems handed down by the ancestral communities of computer science, systems engineering, philosophical logic, and the like (Suchman and Trigg 1992, 174).

While Bayes Theorem dates from the 18th century, the highly successive use of Naive Bayes classifiers in email spam filtering in recent decades effectively draws on an ancestral community of document classification and information retrieval methods.¹²

Early attempts to use what is now called Naive Bayes in the early 1960s re-iterated the engagements with the evidential weight of documents that accompanied the emergence of probabilistic thinking as a quantification of belief in the seventeenth century (Hacking 1975, 35-49). Working at the RAND Corporation in the early 1960s, M.E. Maron described how ‘automatic indexing’ of documents – Maron used papers published in computer engineering journals – could become ‘probabilistic automatic indexing.’ The necessary statistical assumption was:

12. The other lineage descends from medical diagnosis. For instance, starting in 1960, Homer Warner, Alan Toronto and George Veasy, working at the University of Utah and Latter-day Saints Hospital in Salt Lake City, began to develop a probabilistic computer model for diagnosis of heart disease (Warner et al. 1961; Warner, Toronto, and Veasy 1964). Their model used exactly the same ‘equation of conditional probability’ we see in equation 5.3 but now used to ‘express the logical process used by a clinician in making a diagnosis based on clinical data’ (Warner et al. 1961, 177). Despite the mention of logic in this description, the diagnostic model was thoroughly probabilistic in the sense that the model itself has no representation of logic included in its workings. Rather it calculates the probability of a given type of heart disease given ‘statistical data on the incidence of symptoms’ (Warner, Toronto, and Veasy 1964, 558). Somewhat ironically, as they point out, physicians involved in preparing and submitting data to the diagnostic program improved the accuracy in their own diagnoses. In 1964, N.J Bailey was taking the same approach to medical diagnosis (Bailey 1965). Heart disease to a central topic in machine learning (see chapter 4 for discussion of the *South African Heart Disease dataset*).

Maron, M.E.
 machine learner!Naive
 Bayes!history of
 probabilisation!ancestral
 communities of

The fundamental thesis says, in effect, that statistics on kind, frequency, location, order, etc., of selected words are adequate to make reasonably good predictions about the subject matter of documents containing those words (Maron 1961, 406)

This fundamental thesis has remained somewhat fundamental in text classification and information retrieval applications, as well as many other machine learning approaches since. Maron's work focused on a collection of several hundred abstracts of papers published in the March and June 1959 issues of the *IRE Transactions on Electronic Computers*. As in contemporary supervised learning, these abstracts were divided into two groups, a training and a test set ('group 1' and 'group 2' in Maron's terminology (407)), and the training set was classified according to 32 different categories that had already been in use by the Professional Group on Electronic Computers, the publishers of the *IRE Transactions*. Given these classifications, word counts for all distinct words in the abstracts were made, the most common terms ('the', 'is', 'of', 'machine', 'data', 'computer') and the most uncommon words removed, and the remaining set of around 1000 words were actually used for classification. This treatment of the abstracts as documents, then as lists of words, and then as frequencies of terms, and finally as a filtered list of most information rich terms continues in much text classification work today. A typical contemporary information retrieval textbook such as (Manning, Raghavan, and Schütze 2008) devotes a chapter to the topic, including the canonical discussion of how naive assumptions about language and meaning do not vitiate the Naive Bayes classifier. Whenever machine learners affirm the unlikely efficacy of classifiers, we might attend to the ways in which previous 'ancestral probabilisations' of the domain in question prepare the ground for that success.

Statistical decompositions: high bias, low variance

diagram!indexical
errors!bias-variance

and observation of errors

Even with an eye on the ancestral communities that constantly accompany and heavily shape the indexical diagram of machine learning in the world, we still need a way of accounting for the unreasonable artificiality of something like the Naive Bayes. It generates highly arbitrary probabilities of document class membership, yet these arbitrary probabilities allow effective classification. Machine learners view the persistence of manifest artifice (in the case of Naive Bayes, a model that eschews any modelling of relations between things in the word such as words) in terms of another of the structuring differences of machine learning: the so-called *bias-variance_decomposition*(Hastie, Tibshirani, and Friedman 2009, 24). The terms ‘bias’ and ‘variance’ stem from the long history of statistical interest in errors (as Hacking’s account of the transposition of measurement errors into population norms illustrates). The bias and variance of ‘estimators’ – the estimates of the parameters of the models usually written as $\hat{\beta}$ or $\hat{\theta}$ – feature in heavily machine learning discussions of prediction errors. The terms point to tensions that all machine learners experience. On the one hand, *variance* refers to the inevitable reliance of a machine learner on the data it ‘learns.’ To put it more formally, ‘variance refers to the amount by which \hat{f} would change if we estimated it using a different training data set’ (James et al. 2013, 34). On the other hand, *bias* ‘refers to the error that is introduced by approximating a real-life problem, which may be extremely complicated, by a much simpler model’ (35). These two sources of error, one which results from sampling and the other arising from the structure of the model or approximating function, can be reduced or at least subject to trade-off in what *Elements of Statistical Learning* terms ‘the bias-variance

Friedman, Jerome!on
bias-variance
decomposition

decomposition' (Hastie, Tibshirani, and Friedman 2009, 223).¹³ From the standpoint of the bias-variance decomposition, every machine learner makes a trade-off between the errors deriving from differences between samples, and errors due to the difference between the approximating function and the actual process that generated the data. Note that both sources of error in the bias-variance decomposition derive from the relation between the model and the data, one relating to how the model encounters the world (as a set of small samples or as, at the other end, a massive $N = \forall X$ dataset), the other relating to how the model 'sees' the world (as a set of almost coin-toss like independent events, as a geometrical problem of finding a line or curve that runs through a cloud of points, etc.).

Both prediction and inference in machine learning cannot fully circumvent the tensions between the different errors at work in the bias-variance decomposition. Yet, these sources of error do not always prove harmful. The counter-intuitive success of the Naive Bayes and k nearest neighbours classifiers works against the long standing trend in statistics to construct increasingly sophisticated models of the domains they encounter. Writing in 1997, Jerome Friedman describes how very simple classifiers perform surprisingly well:

These effects are seen to be somewhat counter intuitive in both their strength and nature. In particular the bias and variance components of the estimation error combine to influence classification in a very different way than with squared error on the probabilities themselves. Certain types of (very high) bias can be canceled by low variance to produce accurate classification
(Friedman 1997, 55)

13. Another source of error, the 'irreducible error' (Hastie, Tibshirani, and Friedman 2009, 37) is noise that no model can eliminate.

A rather elaborate set of concepts and techniques address the bias-variance decomposition. These techniques focus on managing the *test* or *generalization* error, the difference between the actual and predicted values produced by the machine learner when it encounters a fresh, hitherto unseen data sample. Note that this problem of generalization error is not eliminated in any of the ‘big data’ or $N = \forall X$ scenarios often discussed in association with contemporary information infrastructures. Here too, prediction and inference encounters fresh data in some form or other, and machine learners in such settings still suffer from the bias-variance trade-off. This trade-off has to deal with the fact that training errors – the difference between what the model predicts and what the training data actually shows – is not a good guide to the test or generalization error. The process of fitting a model or finding a function (see previous chapter) will tend to reduce the training error by fitting the function more and more closely to the shape of the training data, but when it encounters fresh data that function might no longer fit well. In other words, a more sophisticated function may well reduce the bias but increase the variance. ‘Richer collections of models’ (Hastie, Tibshirani, and Friedman 2009, 224) reduce bias, but tend to increase variance. Conversely, models that cope well with fresh data (and Naive Bayes is a good example of such a machine learner), display low variance but high bias. The exact nature of the trade-offs between bias and variance shift markedly between different types of models, and generates many different conceptual analyses of error in machine learning literature (‘optimism of the training error rate’ (228), ‘estimates of in-sample prediction error’ (230), ‘Bayesian information criterion’ (233), ‘Vapnik-Chervonenkis dimension’ (237), ‘minimum description length’ (235)) and technical methods of estimating prediction error (‘cross-validation’ (241), ‘bootstrap methods’ (249), ‘expectation-maximization algorithm’ (272), ‘bagging’ (282), or ‘Markov Chain Monte Carlo (MCMC)’ (279)), many of which date from the 1970s

referentiality (e.g. cross-validation (Stone 1974), bootstrap (Efron 1979), expectation-maximization (Dempster, Laird, and Rubin 1977)).

The daunting armory of concepts and methods all respond to the same underlying problem of reference. They invoke in some cases sophisticated mathematical or statistical constructs, but also very often rely on computational iteration to optimise values of parameters in models whose underlying intuitions remain quite simple and straightforward (as in a linear regression or Naive Bayes). In some cases, and Naive Bayes would be a good example, the implementation of a model may be very simple, but analysis of how the machine learner manages to control a source of error such as bias or variance entails much more sophisticated statistical understanding. Many techniques and re-conceptualisations of how a model becomes a ‘useful approximation’ treat the models themselves as a kind of population whose variations and uncertainties, whose tendencies and predispositions must be sampled, tested and monitored.¹⁴ Again, in terms of the structuring differences shown in Table ??, while the model may be predictive in its relation to a given domain or field, the model itself as a machine learner is the object of inference, and sometimes prediction (as in the case of the generalization error) in machine learning.

Does machine learning construct a new statistical reality?

The bias-variance decomposition points to an irreducible tension in the way that machine learning structures differences in the world. Following a broadly Foucaultean line of argument, Hacking proposes that statistical thinking and practice in the nineteenth and early twentieth century on-

¹⁴. The population of machine learners as a plurality requiring management and control forms a key topic of chapter ??

tologically re-configured things in terms of probability distributions (and the Gaussian distribution in particular). What happens in worlds where the statistical treatment of error – the bias-variance decomposition is a shorthand term for this – structures the selection and assessment of machine learners? I have suggested that both the ancestral probabilisation of domains and the statistical decomposition of prediction error come together in statistical machine learning. The referential structure of the bias-variance decomposition includes both tightly bound points and certainly ‘relatively free or unbound points, points of creativity, change and resistance’ (Deleuze 1988, 44), as we saw in the case of the Naive Bayes classifier. It generates highly erroneous probability estimates but performs well as a classifier. The ‘unbound points’ in any diagrammatic process matter greatly to the relations of force at work in a knowledge-power conjunction. The referential power of the models varies, and every attempt to construct a machine learner in a given setting relies either on the re-iteration of ancestral probabilities (that is, prior structuring of experience in conformity with the curve of some probability distribution) or on the many interactive adjustments, redistributions and re-samplings of the world *and* of the models associated with the bias-variance decomposition.

Mayer-Schönberger and Cukier’s argue that having much data or all data ($N = \forall X$) changes the epistemic power of data is a leitmotif in accounts of predictive modelling and analytics during the last decade.

Using all the data makes it possible to spot connections and details that are otherwise cloaked in the vastness of the information. For instance, the detection of credit card fraud works by looking for anomalies, and the best way to find them is to crunch all the data rather than a sample (Mayer-Schönberger and Cukier 2013, 27).

referentiality

Versions of this claim can be found running through various scientific and business settings throughout the 20th century.¹⁵ In certain settings, $N = all$ has been around for quite a while (as for instance, in many document classification settings where the whole corpus of documents have been electronically curated for decades). Mayer-Schönberger and Cukier rightly emphasize that the huge quantities of data sluicing through some contemporary infrastructures support inferences of probabilities (11). Their description of statistical sampling as a concept ‘developed to solve a particular problem at a particular moment in time under specific technological constraints’ (31)] does not, however, accommodate or acknowledge the entanglements of statistical techniques with power and knowledge, and in particular, the biopolitical structuring of differences since the late 18th century. The very possibility of spotting connections or details that might matter itself relies on machine learners that structure differences in specific ways (generative/discriminative, parametric/non-parametric, bias/variance, prediction/inference). Looking for anomalies becomes possible, but only at the cost of machine learners becoming statistical. Whether or not someone uses Naive Bayes, a topic model, neural networks or logistic regression, random variables, probability distributions, error and model selection practices crowd in around and re-configure machine learners, heavily affecting their referentiality. In many ways, the Mayer-Schönberger and Cukier account pays so much attention to the potentials of data accumulation that they cannot easily attend to the question of how machine learners go into the data as people try to ‘spot connections and details’, or of how what is put into the data that goes beyond $N = all$. Here sampling, estimation, likelihoods, and a whole gamut of dynamic relationships between random variables in joint probability distributions reassert themselves amidst a

15. Later chapters of this book will track several instances of having all the data in the sciences, in government and in business in order to show what having all the data entails in different settings.

population of models. The data may not be sampled, but models moving through the high-dimensional common vector spaces opened up by having ‘all’ the data are sampled. Not all machine learners are strictly speaking probabilistic models ¹⁶, but the referential relation of all machine learners to the world has become increasingly statistical by virtue of the both the ancestral probabilisation and the bias-variance decomposition.

Machine learning inhabits worlds that had already absorbed statistical realities at least a century earlier, whether through the social physics of Quetelet, the biopolitical normals of Francis Galton and his regression to the mean (remember that the linear model of regression is probably the basic machine learning model) or later in the probability functions of quantum mechanics in early twentieth century physics. Given a stochastic ontology, machine learning should be understood as a re-inversion. In this inversion, the probability distributions that had become the underlying reality of many different kinds of populations fold back or re-distribute themselves into populations of devices such as machine learners whose variations and uncertainties mesmerize contemporary classificatory and knowledge (scientific and otherwise) practice. Populations of models are sampled, measured, and aggregated in the ongoing production of statistical realities whose object is no longer a property of individual members of a population (their height, their life-expectancy, their chance of HIV/AIDS), but a meta-population of models of populations and their probability distributions.

16. Machine learning textbooks written by computer scientists tend to define probabilistic models more narrowly. As Peter Flach suggests:

Probabilistic models view learning as a process of reducing uncertainty using data. For instance, a Bayesian classifier models the posterior distribution $P(Y|X)$ (or its counterpart, the likelihood function $P(X|Y)$) which tells me the class distribution Y after observing the features values X (Flach 2012, 47)

But whether they are probabilistic in this sense or not, the evaluation and configuring of machine learners irreducibly depends on a statistical treatment of errors and their trade-offs.

Cortes, Corinna|on
pattern recognition
Whitehead, Alfred
North!on pattern
pattern|modes of
togetherness
machine learner!pattern
recognition|seealso|pattern

Chapter 6

Patterns of accumulation and dispersion

Algorithms for pattern recognition were therefore from the very beginning associated with the construction of linear decision surfaces (Cortes and Vapnik 1995, 273-4)

The notion of pattern involves the concept of different modes of togetherness (Whitehead 1956, 195-6).

Should we hold machine learners accountable for their claims to recognise patterns in data?¹ This chapter explores some major derivations of pattern developing in the last decades of the twentieth century, and in particular, how the notion of pattern itself changes as a result. We have seen the

1. Many authors have suggested that algorithms should be the focus of more attention. Mike Savage, in his account of the growth of ‘descriptive assemblages’ based around large scale data mining of transactions, administrative records and social media practice concludes:

It follows that a core concern might be to scrutinize how pattern is derived and produced in social inscription devices, as a means of considering the robustness of such derivations, what may be left out or made invisible from them, and so forth. We need to develop an account which seeks to criticize notions of the descriptive insofar as this involves the simple generation of categories and groups, and instead focus on the fluid and intensive generation of potential (Savage 2009, 171)

abstraction!diagrammatic emergence of the common vector space and its epistopic transformations, machine learner!decision tree the multiplication of functions and their associated partial observers, and then the re-distribution of probability that transform machine learners into populations of error-sensitive learners. On several occasions, problems of the density and mass of techniques, research literature, algorithms and code have appeared. What in this diagram and in the forest-like growth of techniques, projects, applications and proponents allows us to make sense of the dynamics of this accumulation? Across the vectors, functions and distributions, a diagram of machine learning weaves and knots many points of emergence, continuity and conjunction. I view the formidable accumulations of infrastructure, devices and expertise accreting around machine learning as multi-faceted abstractions, where abstraction is understood diagrammatically. Two highly developed and heavily used machine learners – decision trees and support vector machines – more or less mesmerised the machine learning scientific literature between 1980-2000. They initiated relatively novel and somewhat discontinuous diagrammatic movements into data.² These diagrammatic movements, which we might characterise as *splitting*, and *hyper-planing* not only animate machine learners in producing newer techniques (random forests, deep belief nets) and intensifying their application. Perhaps more importantly, they re-open the question of what counts as pattern. If machine learning has no single or unified idea of pattern, then claims that machine learners find hidden patterns, that they traverse the occluded volumes of data, should be evaluated carefully.³

The decision tree and support vector machine appear in the machine learning

2. The other major machine learner of this time, the neural net, will be discussed in chapter ??.

3. *Elements of Statistical Learning* uses the term ‘pattern’ only occasionally. The term appears 33 times there, and mainly in the bibliography. Apart from Brian Ripley’s *Pattern Recognition and Neural Networks* (Ripley 1996), statisticians largely eschew the term. Computer scientists like it more, and particularly in work on the classification of images (see Christopher Bishop *Pattern Recognition and Machine Learning* (Bishop 2006)). Hastie, Tibshirani and Friedman, as statistical machine learners, confine their use of pattern to the term ‘pattern recognition’.

scientific publications of the last three decades (1980-2010) as important novelties. They also, I will suggest, embody a new enunciative modality in which regularity and rarity, accumulation and sparsity do not exist in tension or opposition. Practically, they loom large in various contemporary accounts of machine learning as a way of knowing (for instance, in popular machine learning books such as *Machine Learning for Hackers* (Conway and White 2012) or *Doing Data Science* (Schutt and O’Neil 2013)). The machine learning research published in statistics, computer science, mathematics, artificial intelligence and a swathe of related scientific fields bristles with references to decision trees and support vector machine, as well as neural networks. (As we will see, these techniques themselves ingest substantial supplies of knowledge and practice produced by social sciences, insurance and actuarial practice, and marketing research.) The top 20 most cited publications in the field include Ross Quinlan and Leo Breiman’s papers on decision trees (J. Ross Quinlan 1986; Breiman et al. 1984), Vladimir Vapnik and Corinna Cortes’ support vector machines papers (Vapnik 1999; Cortes and Vapnik 1995), an early textbook written by a computer scientist on machine learning (Mitchell 1997), a textbook and software package on data mining using Java (Witten and Frank 2005); a textbook on pattern recognition dating from the 1970s (Duda, Hart, and Stork 2012), a tutorial on an error control technique (ROC - Receiver Operating Characteristics, first developed by the US military during WWII) and somewhat lower, another well-known textbook, this time on neural networks and pattern recognition (Bishop 2006).

enunciative
modality!seestate-
ment!enunciative
modality of

The re-configuration of tensions between sparsity and accumulation, between rarity and commonality can be understood in the sense of *positivity* proposed by Michel Foucault in *The Archaeology of Knowledge* (Foucault 1972), a book whose somewhat a-subjective approach to knowledge unexpectedly resonates

positivity

Foucault, Michel on
positivity

Foucault, Michel on
statements

with much that takes place in machine learning. A striking *positivity* takes shape across these differences. (I understand positivity partly in the sense of ‘positivism,’ an epistemological stance widely shared amongst machine learners that views knowledge as deriving from observation.⁴) Foucault theorises statements as elements *posited* or deposited in a relatively uneven space of transformations. These transformations generate what can be thought and done within a particular discursive formation. Their positivity arises from the ‘forms of an accumulation’:

To describe a group of statements not as the closed, plethoric totality of a meaning, but as an incomplete, fragmented figure; to describe a group of statements not with reference to the interiority of an intention, a thought, or a subject, but in accordance with the dispersion of an exteriority; to describe a group of statements, in order to rediscover not the moment or the trace of their origin, but the specific forms of an accumulation, is certainly not to uncover an interpretation, to discover a foundation, or to free constituent acts; nor is it to decide on a rationality, or to embrace a teleology. It is to establish what I am quite willing to call a positivity (125).

Foucault frequently refers to positivity in those passages where he discusses how practices develop and how statements relating to those practices take shape. The situations in which he invokes positivity involve accumulations or plethora of statements whose coherence or unity does not lie in any origin, intention, rationality, subjectivity, ideology or teleology. Like mineral strata, positivities entail accumulations without deriving them from underlying unities. Foucault’s acceptance of ‘positivity,’ in spite of the risk of attracting

4. See (Kitchin 2014) for a survey of the major epistemological commitments in ‘Big Data’ discourse. These largely derived from predictive and inferential techniques of machine learners.

criticism, pivots on his interest in, and practical commitment to, understanding accumulations and uneven distributions. A positivity is a way of describing ‘a group of statements’ that may be quite dispersed, discontinuous, and unevenly distributed because they make up ‘a population of events in the space of discourse in general’ (27). If *The Archaeology of Knowledge* is a book concerned with the tissue of relations between things, what is said, what is done and what counts as knowledge (scientific or not) in practice, then the persistent and at times forceful reiteration of *dispersion* and *discontinuity* might seem strangely counterintuitive. But Foucault’s insistence on discourse as something irreducible to logic, language, intention, rationality, experience, historical origin or narrative, and as the way in which ‘statements’ (‘a graph, a growth curve, an age pyramid, a distribution cloud are all statements’ (82)), strategies, concepts, objects, archives and knowledges come together in ‘discursive formations’ is an attempt to resist any resort to a hidden order or concealed origin that critical thought could uncover or interpret.

positivity!accumulation
in
statement!as diagram

Positivity as accumulation helps make sense of a field such as machine learning whose many different forms, techniques, statements, institutions, values and associations often grapple with problems of much data, many variables, many possible models and a sense of an elusive but vital order or pattern discoverable amidst accumulation. Rather than seeing pattern as something discovered in data, the notion of positivity suggests we should examine the diagrammatic operations that configure the practice of machine learning, giving rise to a field of objects and subject positions. The three machine learners that anchor this chapter are at once perhaps the most distinctive data mining, pattern recognition and predictive modelling achievements of the late twentieth century (at least judging by the citational and implementational interest they attract). They different greatly

statement!enunciative
modality of
statement!enunciative
function of

in how they move through data. At certain times, they come together (for instance, in machine learning competitions discussed in chapter 8; or in certain formalizations such as machine learning theory or in graphs of the bias-variance decomposition discussed in chapter 5; or in the pedagogy of machine learning discussed in chapter 2). We might understand their differences in terms of ‘enunciative modalities’ (54) – that is, in terms of the different ways in which they give rise to statements. Every machine learner generates statements, but from different places, by somewhat different individuals, and from the different situations they that are able ‘to occupy in relation to the various domains or groups of objects’ (52).⁵

Splitting or recursive binary partitioning and the growth of trees

Mastering the details of tree growth and management is an excellent way to understand the activities of learning machine generally (Malley, Malley, and Pajevic 2011, 118).

Decision trees promise an understanding of machine learning. The enunciative function of the decision tree partly relates to the observability and comprehensibility of machine learning. As we will see, not all machine learners readily support observation or comprehension. The cost of comprehensibility, however, is a certain highly restricted movement through the data, a movement that although simple and easily understandable, has been difficult to stabilise and control. As *Elements of Statistical Learning* puts it: ‘tree-based methods partition the feature space into a set of rectangles, and

5. While Foucault tends to retain a decoupled subject-object relation in the production of statements, I tend to see these enunciative modalities as distributed across people and things. As always, machine learner is a composite term for this distribution.

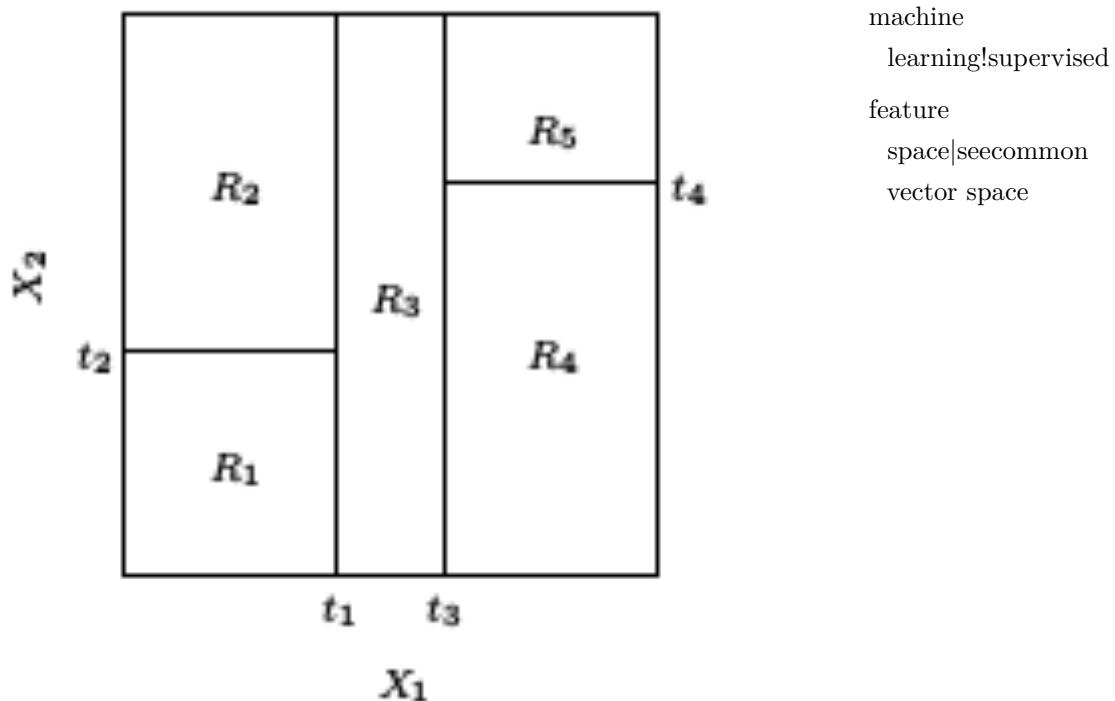


Figure 6.1: Recursive partitioning of the feature space

then fit a simple model (like a constant) in each one. They are conceptually simple yet powerful' (Hastie, Tibshirani, and Friedman 2009, 305).

Tree-based methods are supervised learners as they require the data to either be labelled with a class or to have some outcome value. The variable types in the feature space (or common vector space of the data) can be mixed. Because the method cuts the common vector space into a tiled surface, the features or data variables can be continuous or discontinuous. The 'simple models' that tree methods construct each inhabit one of the rectangular regions or partitions of the feature space. In Figure 6.1, the different regions or partitions produced by a decision tree are labelled R_1, R_2 etc.

Work on classification and regression techniques using decision trees goes back to the early 1960s when social scientists James Morgan and John Sonquist at the University of Michigan's Institute for Social Research were

machine learner!decision tree!history of machine learner!automatic interaction detector	Title
	Design Of Sample Surveys In Education
	Remedial Research Approaches And Models For Research In Education
	The Achievement Motive And Economic Behavior
	Election Simulation
	Simplification Of Economic Models
	Recent Events In Population Control
	Data Dredging Procedures In Survey Analysis
	Aid Computer Programme, Used To Predict Adoption Of Family Planning In Koyang
	Advertising Performance As A Function Of Print Ad Characteristics
	World Affairs Information And Mass Media Exposure

Table 6.1: References to Morgan and Sonquist's Automatic Interaction Detector

attempting to analyse increasingly large social survey datasets (Morgan and Sonquist 1963). As Dan Steinberg describes in his brief history of decision trees (Steinberg and Colla 2009, 180), the ‘automatic interaction detector’ (AID) as it was known, sought to automate the practice of data analysts looking for interactions between different variables. The variety and sheer optimism of subsequent applications of these prototype decision tree techniques is striking. In the 1960s and 1970s, papers that drew on the AID paper or use AID techniques can be found, as table 6.1 shows, in education, politics, economics, population control, advertising, mass media and family planning.

A decade after initial work, the AID was the object of trenchant criticism by statisticians and others, not for the classifications it used (see Figure 6.2), but for its dangerous empiricism. Writing in the 1970s, statisticians in the behavioural sciences such as Hillel Einhorn at the University of Chicago castigated the use of such techniques. The criticisms stemmed from a

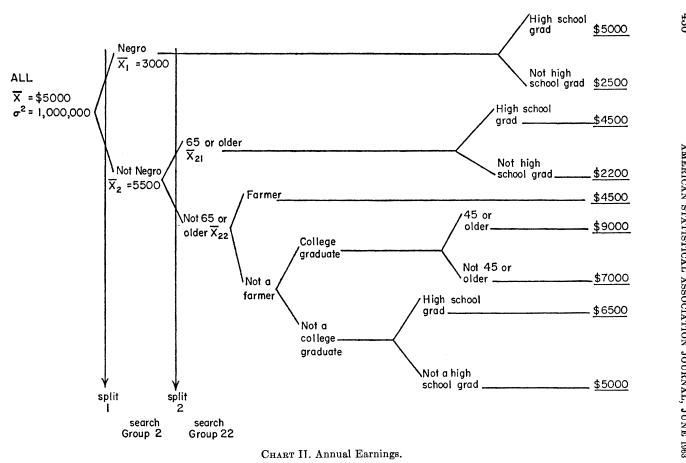


Figure 6.2: AID classifies annual earnings (Morgan & Sonquist, 1963, 430)

general distrust of ‘purely empirical methods’, and scepticism focused on their positivity:

The purely empirical approach is particularly dangerous in an age when computers and packaged programs are readily available, since there is temptation to substitute immediate empirical analysis for more analytic thought and theory building. It is also probably too much to hope that a majority of researchers will take the time to find out how and why a particular program works. The chief interest will continue to be in the output—the results—with as little delay as possible (Einhorn 1972, 368)

Einhorn discusses AID alongside other techniques such as factor analysis and multi-dimensional scaling (both still widely used) before concluding ‘it should be clear that proceeding without a theory and with powerful data analytic techniques can lead to large numbers of Type I errors’ (378). His statistical objections to AID are particularly focused on the problematic power of the technique: ‘it may make sense out of “noise”’ (369). Consequently,

operations research!
use
of decision trees
machine
learner!
overfitting

researchers easily misuse the technique: they ‘overfit’ the data, and do not pay enough attention to issues of validation (369-370). Similarly the British marketing researcher Peter Doyle, criticising the use of AID in assessing store performance and site selection by operations researchers, complained that searching for patterns in data using data sets was bound to lead to spurious results and the decision trees, although intuitively appealing (that is, they could be easily interpreted), were afflicted with arbitrariness: ‘a second variable may be almost as discriminating as the one chosen, but if the program is made to split on this, quite a different tree occurs’ (Doyle 1973, 465-466).⁶ Most of these criticisms can be seen as expressing conventional statistical caution in response to threats to validity.

1984: Cutbacks on recursive partitioning

As Einhorn expected, it was too much to hope that all researchers would take time to investigate how a particular program works. Some researchers however did take time, years in fact, to investigate how decision trees work. Writing around 2000, Hastie, Tibshirani and Friedman, who could hardly be accused of not understanding decision trees, happily recommend decision trees as the best ‘off-the-shelf’ classifier: ‘of all the well-known learning methods, decision trees comes closest to meeting the requirements for serving as an off-the-shelf procedure for data-mining’ (Hastie, Tibshirani, and Friedman 2009, 352). We might wonder here, however, whether they damn with faint praise, since ‘off-the-shelf’ suggests pre-packaged, and commodified, and the term ‘data-mining’ itself is not without negative connotations. As for its commercial realities, in 2013, Salford Systems, the purveyors of the leading contemporary commercial decision tree software,

6. These objections and resistances to early decision trees echo today in discussions around pattern recognition, knowledge discovery and data-mining in science and commerce. The problem of what computers do to the analysis of empirical data is long-standing.

CART, could claim:

CART is the ultimate classification tree that has revolutionized the entire field of advanced analytics and inaugurated the current era of data mining. CART, which is continually being improved, is one of the most important tools in modern data mining. Others have tried to copy CART but no one has succeeded as evidenced by unmatched accuracy, performance, feature set, built-in automation and ease of use. [Salford Systems](#)

machine learner!CART
data mining!in 1970
machine learner!decision tree!in medicine
Friedman, Jerome!work on decision tree

What happened between 1973 and 2013? Decision trees somehow grew out of the statistically murky waters of social science departments and business schools in the early 1970s to inaugurate the ‘current era of data mining’ (which the scientific literature indicates starts in the early 1990s). This was not only a commercial innovation. As the earlier citation from U.S. National Institutes of Health biostatisticians Malley, Malley and Pajevic indicates, decision trees enjoy high regard even in biomedical research, a setting where statistical rigour is highly valued for life and death reasons. The happy situation of decision trees four decades on suggests some kind of threshold was crossed in which the epistemological, statistical, or algorithmic (‘built-in automation’) power of the technique altered substantially.

The third author of *Elements of Statistical Learning*, Jerome Friedman, worked at the U.S. Department of Energy’s Stanford Linear Accelerator during the late 1970s. Friedman was instrumental in rescuing decision trees from the ignominy of profligate ease of use and pure empiricism they had endured since the late 1960s. The reorganisation and statistical retrofitting of the decision tree was not a single or focused effort. During the 1980s, statisticians such as Friedman and Leo Breiman renovated the decision tree as a statistical tool (Breiman et al. 1984) at the same time as computer

Quinlan, John Ross!
 artificial
 intelligence!rule-based
 induction
 vectorization
 machine learner!CART

Breiman, Leo!CART
 monograph

scientists such as Ross Quinlan in Sydney were re-implementing decision trees guided by an artificial intelligence-based formalisation as rule-based induction technique (J. Ross Quinlan 1986).⁷ This uneasy parallel effort between computer science and statistics still somewhat strains relations in machine learning today. Both statisticians and computer scientists do and use the same techniques, but often with the computer scientists focusing on optimisation and algorithmic scale and the statisticians inventing novel statistical formalizations and abstractions. The fateful embrace of statistics and computer science, the disciplinary binary that vectorizes machine learning, has been generative in the retrieval of the decision tree.

An initial symptom of the transformation of the technique appears in a name change. The term ‘decision tree,’ although still widely used in the research literature and machine learner parlance was replaced by ‘classification and regression tree’ during the late 1970s and 1980s. The terms ‘classification and regression tree’ is sometimes contracted to ‘CART,’ and that term strictly speaking refers to a computer program described in (Breiman et al. 1984) as well as the title of that highly-cited monograph. As we have seen in previous chapters, regression and classification designate the two main sides of machine learning practice, and their concatenation with ‘tree’ attests to the ongoing renovation of existing machine learning approaches with a more statistical facade.

The implementation of machine learning techniques in R by virtue of R’s statistical provenance perhaps favours the statistical side of decision tree practice, but that has certain forensic virtues not offered by commercial

7. Quinlan’s papers and book on versions of the decision tree (ID3 and c4.5) are both amongst the top ten the most highly cited references in the machine literature itself. Google Scholar reports over 20,000 citations of the Quinlan’s book *C4.5: Programs for Machine Learning* (John Ross Quinlan 1993) (although far fewer appear in Thomson Reuters Web of Science). Several years ago, C4.5 was voted the top data mining algorithm (Wu et al. 2008). While I don’t discuss Quinlan’s work in much detail here, we should note as a computer scientist, Quinlan takes a much more rule-based approach to decision tree than Breiman and co-authors.

or closed-source software often produced by computer scientists. In this case, the name of one long-standing and widely-used R package itself attests to something: `rpart` is a contraction of 'recursive partitioning' and this term generally describes how the decision tree algorithm works to partition the common vector space into the form shown in Figure 6.1 (Therneau, Atkinson, and Ripley 2015). 'CART,' on the other hand, is a registered trademark of Salford Systems, the software company mentioned above, who sell the leading commercial implementation of classification and regression trees. Hence, the R package `rpart` cannot call itself the more obvious name `cart`, and instead invokes the algorithmic process it relies on: recursive partitioning.⁸

algorithm!recursive
partitioning
programming
languages!R!packages!rpart
dataset!iris

```
data(iris)
library(rpart)
iris_tree = rpart(Species ~ ., iris)
```

R.A. Fisher's *iris* dataset, which contains measurements made in the 1930s of petal and sepal lengths of *iris virginica*, *iris setosa* and *iris versicolor* is a standard instructional example for decision trees (R. Fisher 1938).⁹ The code shown here loads the *iris* data (the dataset is routinely installed with many data analysis tools), loads the `rpart` decision tree library, and builds a decision to classify the irises by species. What has happened to

8. Other R packages such as `party` (Hothorn, Hornik, and Zeileis 2006) and `tree` (Ripley 2014) also use recursive partitioning, but with various tweaks and optimisations that I leave aside here.

9. `iris` is a very small dataset, a pre-computational miniature. That diminutive character makes it diagrammatically mobile. It supports a rhizomatic ecosystem of examples scattered across the machine learning literature. The usual framing of the classification problem is how to decide whether a given iris blossom is of the species *virginica*, *setosa* or *versicolor*. These irises don't grow in forests – they are more often found in riverbanks and meadows – but they do offer a variety of illustrations of how machine learning classifiers are brought to bear on classification problems. Here the classification problem is taxonomic - the iris genus has various sub-genera, and sections within the sub-genera. `_Setosa`, `virginica` and `versicolor` all belong to the sub-genus *Limniris*. This botanical context is routinely ignored in machine learning applications. In machine learning textbooks and tutorials, `iris` typically would be used to demonstrate how cleanly a classifier can separate the different kinds of irises.

code!brevity of!
data!dimensionality
machine learner!function

the iris data in this decision tree? The R code that invokes the recursive partitioning algorithm is so brief `iris_tree =rpart(Species ~ ., iris)` that we can't tell much about how the data has been 'recursively partitioned.'

We know that the *iris* has 150 rows, and that there are equal numbers of the three iris varieties.

Code brevity indicates a great deal of formalization of practice has accrued around decision trees. Some of this formalization was described in the landmark *Classification and Regression Trees* monograph (Breiman et al. 1984), and is familiar (from previous chapters) in its use of common vector space, functions as partial observers and the re-distribution of chance between devices and worlds. For instance, Breiman and co-authors start out by placing the technique in vector space:

Define the measurements (x_1, x_2, \dots) made on a case as the *measurement vector* \mathbf{x} corresponding to the case. (3)

In re-configuring decision trees as classification trees, Breiman first addresses issues of data dimensionality. Next, rather than simply invoking the notion of a classifier as a device (as AID did), the CART monograph defines a classifier in terms of a function:

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

Note that 'rule,' the term that computer scientists working in artificial intelligence at the time were using, still figures here, but the classifier is formalized as a function. The definition of the classifier as a function depends on the existence of a vector space \mathcal{X} , a set of measurement, feature or predictor vectors \mathbf{x} and a set of responses $1, 2, \dots, J$. Second, classification is

understood in terms of a series of binary splits in the different dimensions of the common vector space. These splits comprise the tree structure diagram.

The problem here is that many splits are possible.

difference!Gini index of diversity

learning!as dividing

The first problem in tree construction is how to use \mathcal{L} to determine the binary splits of \mathcal{X} into smaller pieces. The fundamental idea is to select each split of a subset so that the data in each of the descendant subsets are “purer” than the data in the parent subset (23).

Tree construction hinges on the notion of purity or more precisely ‘node impurity’, a function that measures how data labelled as belonging to different classes are mixed together at a given branch or node in a decision tree: ‘that is, the node impurity is largest when all classes are equally mixed together in it, and smallest when the node contains only one class’ (24). As Malley and co-authors note, ‘the collection of purity measures is still a subject of research’ (Malley, Malley, and Pajevic 2011, 123), but Breiman, Friedman, Olshen and Stone promoted a particular form of impurity measure for classification trees known as ‘Gini index of diversity’ (Breiman et al. 1984, 38). Like the planar decision surface used in machine learners based on linear regression models, recursive partitioning combined with measures of node impurity allow tree construction to be understood as a form of cutting, division or classification Whereas in linear model-based machine learners, the intuition motivating the function-finding or learning was ‘find the line that best fits or separates the data, index{machine learner!logistic regression!gradient descent} here the intuition is more like: ’split things in ways that minimize mixing’. Good splits decrease the level of impurity in the tree. In an tree with maximum purity, each terminal node – the nodes at the base of the tree – would contain a single class.

dataset!iris

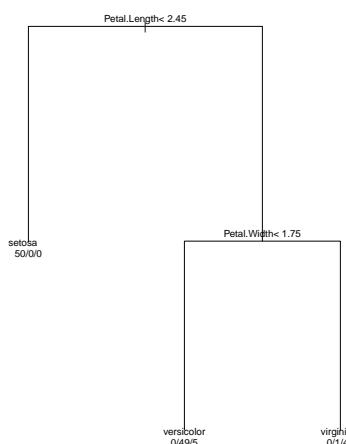
The results of the application of measures of node impurity can be seen below in two plots.

setosa versicolor virginica

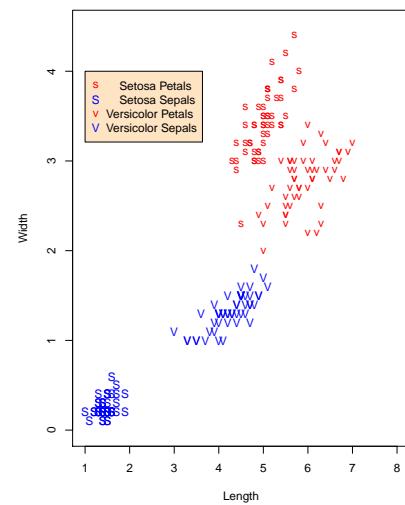
50 50 50

\begin{figure}

tree recursive partitioning of iris



Petal and Sepal Dimensions in Iris Blossoms



\caption{Decision tree on \verb{iris} dataset} \end{figure}

In Figure 6, the plot on the left shows the decision tree and the plot on the right shows just *setosa* and *versicolor* plotted by petal and sepal widths and lengths. As the plot on the right shows, most of the measurements are well clustered. Only the *setosa* petal lengths and widths seem to vary widely. All the other measurements are tightly bunched. This means that the decision tree shown on the left has little trouble classifying the irises. Decision trees are read from the top down, left to right. The top level of this tree can be read, for instance, as saying, if the length of petal is less than 2.45, then the iris is *setosa*.

Like logistic regression models, neural networks, support vector machines or any other machine learning technique, decision trees make sense of the

worlds they encounter in terms of specific qualities and logics. Recursive partitioning can split and sub-divide the common vector space to capture every minor difference between cases, and thereby achieve a perfect fit to the data. Although the partitioning or splitting rules have strong statistical

algorithm!recursive
partitioning
error!variance
error!overfitting

justifications, they do not at all eliminate the problem of instability or variance in trees. For instance, they easily end up ‘overfitting’ the data. Overfitting is a problem for all machine learning techniques. Algorithms sometimes find it hard to know when to stop. During construction of a decision tree, various features in the data are split into smaller and smaller groups. ‘The goodness of the split’, wrote Breiman and co-authors, ‘is defined to be the decrease in impurity’ (25). Under this definition of goodness, the terminal nodes or leaves of the tree can end up containing a single case, or a single class of cases. The decision tree targets the

singularity of the individual case to such a degree that it sees categorical differences everywhere. Driven to maximise the purity of the partitions it creates, it leans too heavily on data it has been trained on to see relevant similarities when fresh data appears. Trees that branch too much are too sensitive (or unstable), and need to be pruned. Such a model will almost always *overfit*, since slight variations in the values of variables in a fresh case are likely to yield widely differing predictions. In the terminology of machine learning, such a decision tree will have low bias but high variance.

Much of the development of decision tree did not revolve around how to construct them, but how to limit their growth. An associated literature on *pruning* decision trees using measures of tree complexity has also developed. As Hastie and co-authors put the problem in their account of classification and regression trees:

How large should we grow the tree? Clearly a very large tree might overfit the data, while a small tree might not capture the

important structure. Tree size is a tuning parameter governing the model's complexity, and the optimal tree size should be adaptively chosen from the data. One approach would be to split tree nodes only if the decrease in sum-of-squares due to the split exceeds some threshold. This strategy is too short-sighted, however, since a seemingly worthless split might lead to a very good split below it. The preferred strategy is to grow a large tree T_0 , stopping the splitting process only when some minimum node size (say 5) is reached. Then this large tree is pruned using cost-complexity pruning (Hastie, Tibshirani, and Friedman 2009, 307-308).

Growing a maximum decision tree, and then cutting back its branches using a cost-function is a further renovation of the decision tree device as a machine learner. The 'cost complexity pruning' reproduces the optimisation we have already discussed in relation to linear regression and logistic regression models. As in these techniques, the definition of a cost function controlling the 'complexity' of a tree – how many branches and leaves/nodes it contains, combined with measures of how well it classifies or predicts – opens up the possibility of iteratively testing different versions of tree against each other. 'We define the cost complexity criterion,' write

Hastie and co-authors, as:

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha |T| \quad (6.1)$$

The idea is 'to find, for each α , the subtree $T_\alpha \subseteq T$ to minimize $C_\alpha(T)$ ' (308). For present purposes, we need only recognise that the cost complexity function re-configures decision trees (T in equation 6.1) as

optimisation problems, in which the variation of a parameter (α) allows minimization of a derived value (the cost C_α).

function!cost

referential

diagram!decision tree as

The shift from AID to CART enunciates a change in how patterns become visible. The decision tree algorithm combines recursive partitioning and cost-complexity pruning to configure a different mode of enunciation, a ‘specific form of accumulation’ (Foucault 1972, 125). When they cross such thresholds, machine learners create a new *referential*, a new field of rules of differentiation of individuals, facts, things and relations.¹⁰ The new referential – the combination of purity and density that comes from recursive partitioning and cost-complexity pruning – re-defines what counts as pattern. Hence, classification trees stand in a different relation to classification, and prediction than their predecessor diagrams. While the graphic form of the decision tree was, by virtue of the long-standing diagrammatic practice of tree-drawing, easy to interpret, observation of decision trees had no way of gauging the instability or variability of any given tree. Hastie and co-authors write: ‘one major problem with trees is their high variance. Often a small change in the data can result in a very different series of splits, making interpretation somewhat precarious. The major reason for this instability is the hierarchical nature of the process: the effect of an error in the top split is propagated down to all of the splits below it (Hastie, Tibshirani, and Friedman 2009, 312). The very diagrammatic form that allows decision trees to be observed and interpreted is also the source of their instability.

If decision trees have been heavily used in credit risk assessment as well as many biomedical models, does this stem from the visibility of the patterns

10. Foucault assigns all statement to referentials. He states that ‘the referential of the statement forms the place, the condition, the field of emergence, the authority to differentiate between individuals or objects, states of things and relations that are brought into play by the statement itself; it defines the possibilities of appearance and delimitation of that which gives meaning to the sentence, a value as truth to the proposition’ (Foucault 1972, 91).

enunciative modality

machine learner!random
forest

pattern!operational

pattern!in dispersion

they produce, even if those patterns are unstable? Or is the success of the decision tree perhaps better understood as a change in the referential underpinning of patterns more generally, of which decision trees would only be one instance among many? If we understand machine learners as enunciative modalities (remembering that the concept of an enunciative modality concerns the dispersion and discontinuities that surround, support and actually generate statements), the transformation and re-modelling of the decision tree as classification and regression trees suggests a subtle, non-localizable discontinuity. We could follow here the subsequent development of the decision tree and its subsequent transmogrification into random forests (L. Breiman 2001), that grow a myriad of small decision trees. Their growth and inter-comparison disperses kaleidoscopic fragments of classificatory order without unifying pattern. They intensify the ‘discontinuity of the planes’ from which machine learners speak.¹¹ In such developments – and we could also consider here techniques, models and methods of ‘boosting,’ ‘bagging,’ or ‘ensemble learning’ that conducts ‘supervised search in a high-dimensional space of weak learners’ [Hastie_2009, 603], pattern has an operational rather than visible mode of togetherness.

The successful dispersion of the support vector machine

Already in the growing and pruning of the decision tree, and even more markedly in support vector machine and neural networks, a different kind of enunciative function can be discerned in which patterns plays out in dispersion and discontinuity rather than in regular geometry. Take the case

11. Chapter 8 discusses subject positions and machine learning. The current discussion concerns changes in referentiality associated with machine learning.

of the support vector machine. The second most highly cited reference in the last few decades of machine learning literature is a paper from 1995 by Corinna Cortes and Vladimir Vapnik of AT & T Bell Labs in New Jersey, USA entitled ‘Support Vector Networks’ (Cortes and Vapnik 1995). Few women’s names have appeared prominently in the machine learning literature. The computing science and statistics departments at Stanford and Berkeley, the laboratories at Los Alamos and AT&T Bell between the 1960s and the 1980s were, it seems, not overly popular or populated with women scientists and engineers. Some prominent machine learning researchers at the time of writing are women (I return to this in chapter 8), but Cortes is perhaps pre-eminent both as head of Google Research in New York (2014) and as recipient with Vapnik of an Association for Computing Machine award in 2008 for work on the support vector machine

Cortes, Corinna
Vapnik, Vladimir
support vector
machine|seemachine
learner!support vector
machine

algorithm.¹²

The rapid rise to popularity of the support vector machine can be seen in the machine learning research literature, very small slice of which appears in Table ???. A substantial fraction of the overall research publication since the mid-1990s accumulates around this single technique, and as usual ranges across credit analysis, land cover prediction, protein structures, brain states and face recognition. The support vector machine covers the normal biopolitical gamut of life, labour and language. The influence of the technique can also be seen in adjacent and overlapping fields such as pattern recognition and data mining, where (Cortes and Vapnik 1995) and similar papers rank near the top-cited papers.¹³ This kind of growth betokens high levels of interest, identification and investment on the part of

12. The support vector machine is distinctive in its mode of movement, and without being swamped by the technical details, we might grasp something of this from Vapnik's writings. Vapnik trained and worked of decades in the former USSR as a mathematician and statistician. His writings on the problems of pattern recognition contrast greatly with other engineers, statisticians and computer scientists in their robustly theoretical formalism. A highly cited 1971 publication with Alexey Chervonenkis 'On the uniform convergence of relative frequencies of events to their probabilities' (published in Russian in 1968) (Vapnik and Chervonenkis 1971) sets the formal tone of this work. In ensuing publications in Russian and then in English after Vapnik moved from Moscow to AT&T's New Jersey Bell Labs in 1990, Vapnik's work remains quite formally mathematical. Although it pertains to 'learning machines,' machine here are understood mathematically simply as 'the implementation of a set of functions' (Vapnik 1999, 17). The way that Vapnik develops a theory of learning owes little visible debt to actual attempts to work with data or experience in doing statistics in any particular domain. This contrasts greatly for instance with the work of statisticians like Breiman or Friedman or even computer scientists like Quinlan or Le Cun whose work lies much closer to fields of application. Vapnik's work, like that of the Russian mathematician Andrey Kolmogorov he draws on, differs from many other contributions to machine learning partly by virtue of this formality and its efforts to derive insight into machine learning by theorising learning. The *Vapnik-Chervonenkis dimension*(VC dimension), a very widely used way of defining the capacity of a particular machine learning technique to recognise patterns in data dates from his work in the 1960s and underpins a general theory of 'learning.' Vapnik writes in 1995,

The VC dimension of the set of functions (rather than the number of parameters) is responsible for the generalization ability of learning machines. This opens remarkable opportunities to overcome the "curse of dimensionality" (83).

As we will see in this chapter, Vapnik's attempts to overcome dimensionality also re-shape what counts as pattern.

13. *Elements of Statistical Learning* also devotes a chapter to support vector machines (Hastie, Tibshirani, and Friedman 2009, Chapter 14)

referential!dispersed

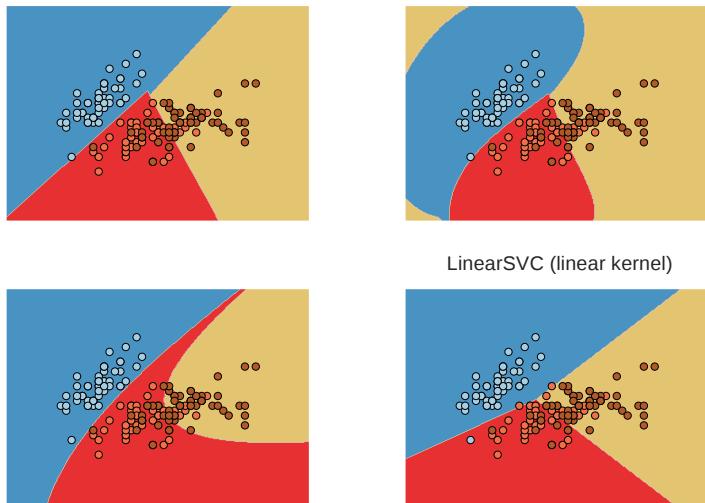


Figure 6.3: SVM on Anderson's iris dataset

the researchers, and presumably more widely. What is it about this technique?

The shift from decision trees to classification tree (and then random forests) illustrate an enunciative modality anchored in a dispersed referential. The support vector machine shown in Figure 6.3 demonstrates perhaps more strikingly how this dispersed referential changes what counts as pattern.

The typical regions of the different sub-graphs have different contours, contours that project a much more dispersed referential. While the name is somewhat forbiddingly technical compared to more familiar terms such as ‘decision tree’ or even ‘neural network,’ the underlying intuition of the technique is much older, and can be found in the models developed by the British statistician R. A. Fisher during the 1930s. R.A Fisher developed the ‘first pattern recognition algorithm’ (273), the ‘linear discriminator function’ (R. A. Fisher 1936), to deal with problems of classification, and demonstrated its efficacy on the taxonomic problem of discriminating or

dataset!iris
 function!linear
 discriminant
 Fisher, Ronald Ayre
 classification!decision
 boundary
 function!discriminant
 pattern!dispersion of

classifying the irises observed in W.E. Anderson's irises in `iris` dataset (see above). In his 1936 article in the *Annual Review of Eugenics*, Fisher comments on similar classification work carried out in craniometry and other related settings: so-called 'discriminant functions' had been successfully used to distinguish populations. Fisher wrote: 'when two or more populations have been measured in several characters, ... special interest attaches to certain linear functions of measurements by which the populations are best discriminated' (179). The discriminant functions divide the common vector space into 'a collection of regions labeled according to classification' (Hastie, Tibshirani, and Friedman 2009, 101). 'Decision boundaries' (or sometimes 'decision surfaces') often appear as straight lines that divide the common vector space into regions of constant classification. The route by which these long-standing discriminant functions were reconstructed during the 1990s in the form of the support vector machine highlights shifts in practice that give rise to new statements about data, statements that can be glimpsed in table ?? in the range of 'referents,' things, facts and beings running through the titles of the papers.

The decision boundary blurs?

The linear decision boundary changes in two ways in support vector machines. It blurs and bends. Nearly all expositions of the support vector machine including (Cortes and Vapnik 1995) highlight the 'soft margin' that runs in parallel to the solid decision boundary. The support vector machine addresses the problem of how to model differences when differences are blurred. An oft-repeated illustration of how the support vector machine traverses data appears in Cortes and Vapnik's initial publication simply entitled 'Support Vector Networks' (Cortes and Vapnik 1995). They demonstrate how the support vector machine classifies

handwritten digits drawn from a dataset supplied by the US Postal Service (LeCun and Cortes 2012). Like `iris`, the US Postal Service digits and a larger version from the US National Institute of Standard (NIST) are standard machine learning dataset. They have been frequently used to measure the performance of competing learning algorithms. In contrast to *iris*, the US Postal Service Database is high dimensional. Each digit in the dataset is stored as 16x16 pixel element. Image classification typically treats each pixel as a feature or variable in the input space. So each digit as represented by a 16x16 pixels amounts to a 256 dimensional input space. By comparison, `iris` has five dimensions. Unsurprisingly, there are also many more digits in the US Postal Service Database than in flowers in *iris*. The NIST dataset has around 70,000. Aside from this dimensional growth, the handwritten digits aptly convey the non-linearity of the classification problem. On the one hand, human can recognise handwritten digits fairly easily and with few errors. This is despite the many variations in handwriting that skew, morph and distort the ideal graphic forms of numbers.¹⁴

In their experiments with digit recognition (shown in figure 6.4, Cortes and Vapnik contrast the error rates of decision trees (CART and C4.5), neural networks and the support vector machine working at various level of

14. Neural network researchers have heavily used the MNIST dataset. I discuss some of that work in chapter 8. The handwritten MNIST also appear in *Elements of Statistical Learning*, where they are used to compare the generalization error (see previous chapter) of a k nearest neighbours, convolutional neural network, and a ‘degree-9 polynomial’ support vector machine (408). What about the handwritten digits attracts so many machine learning techniques? The logistics of the US Postal Service aside (since the MNIST datasets continue to be used by machine learners well after the problem of scrawl on letters has been sorted), the variations, the regularities, and the banal everydayness of these digits furnish a referential locus, whose existence as a facts, things or events in the world is less important than the relations of similarity and differences it poses. The field of digitals becomes a site of differentiation not only of digits – the machine learners attempt to correctly classify the digits – but of the authority of different machine learning techniques and approaches. They become ways of announcing and delimiting the authority, the knowledge claims or ‘truth’ associated with the machine. The many uses of the MNIST data documented by (LeCun and Cortes 2012) suggests something of the referential force of such datasets.

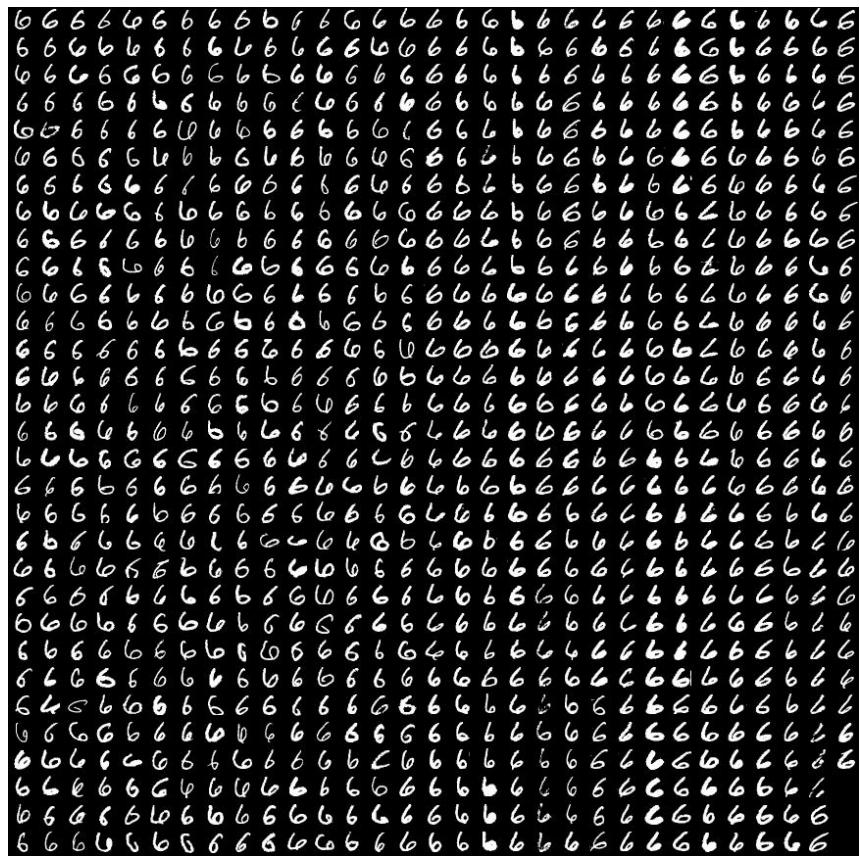


Figure 6.4: MNIST Postal Digits

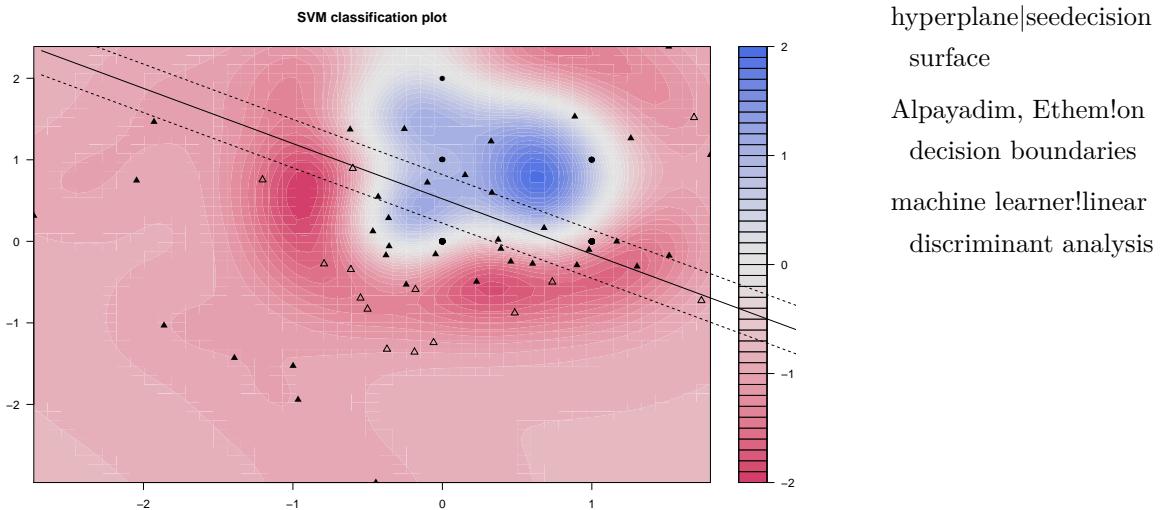


Figure 6.5: Margins in a support vector machine

dimensionality. This occurs in two different ways. On the one hand, the support vector machine develops Fisher's linear discriminant analysis since it searches for a separating hyperplanes in the data. While linear discriminant analysis constructs that hyperplane by finding the most likely boundary between classes based on all the data, the support vector machine searches for a hyperplane resting only on those cases in the data that lie near the boundary. It introduces the intuition that the best hyperplane separating different classes will run near to the cases – the *support vectors* – that are most difficult to classify. These hard-to-classify cases become the support vectors whose relative proximities tilt the decision surface in various directions. In contra-distinction to the $N = \forall X$ proposition (discussed in chapter 5), the machine learner discards much of the data. As Alpayadin writes, ‘we do not care about correctly estimating the densities [probability distributions of variables] inside class regions; all we care about is the correct estimation of the *boundaries* between the class regions’ (Alpaydin 2010, 210).

Figure 6.5 has appeared in many slight variations in the last two decades.

hyperplane|seed
surface
Alpayadim, Ethem!on
decision boundaries
machine learner!linear
discriminant analysis

Such figures diagram classes by different point shapes, and the diagrammatic work of the classifier takes the shape of diagonal lines, the solid line marking the decision surface or hyperplane and the dotted lines marking the soft margins that separate the two classes. In Figure 6.5, the dotted lines represent a margin on either side of a hyperplane (the solid line). The support vector machine finds the hyperplane for which that margin or perpendicular distance between the margins is greatest. Of all the slightly different planes that might run between the two classes shown in that figure, the maximum separating hyperplane lies at the greatest distance from all the points of the different classes. The support vector classifier modifies the idea of the optimal separating hyperplane by allowing classification of inseparable or overlapping classes.

While the geometrical intuition here is that some cases will be allowed to lie on the opposite side of the decision surface to where they should be, the distance they lie on the wrong side of the separating hyperplane will be as small as possible. How are the lines showing in Figure ?? calculated?

Combining the optimal separating hyperplane and a limited number of permitted misclassifications entails a complicated optimisation problem. As Hastie and co-authors, following Cortes and Vapnik's formulations, formalize it, the problem can be stated as a minimization problem:

$$L_P = \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^N x_i - \sum \alpha [y_i(x_i^T \beta + \beta_0) - (1 - x_i)] - \sum_{i=1}^N \mu_i x_i \quad (6.2)$$

(Hastie, Tibshirani, and Friedman 2009, 420)

In equation 6.2, the optimisation problem is to minimize L_P , the Lagrange primal function with respect to β, β_0 and x_i (420). In this complicated optimisation problem (one that is difficult to understand without extensive

mathematical background), the key elements are the now familiar β parameters in the linear form of $x_i^T \beta + \beta_0$, which is the equation defining the hyperplane, as well as the multiple occurrences of sums (Σ) of all the values x_i , which calculate the distance that each case is on the wrong side

of the margin. Correctly classified cases will, therefore, have $x_i = 0$. As always with these mathematical propositions, their diagrammatic relations, and the way in which they contain both the generalizing regularities (the

linear equation, the repeated summation of all values, the shaping parameters) and forms of variation (the presence of the misclassification measures x_i) should be the focus of our attention. What if elements that lie on the wrong side of the hyperplane were allowed? If that were possible, the support vector machine could deal with in-separable or overlapping classes. As Hastie and co-authors put it in *Elements of Statistical Learning*, ‘hence we are able to find the hyperplane that creates the biggest margin between the training points for class 1 and -1’ (418). The support vector machine permits instances that lie on the wrong side of the separating hyperplane. They function not as errors (as they would appear in most linear classifiers such as linear discriminant analysis and logistic regression),

but as components of a ‘soft margin’ designed to accommodate inseparability and indistinctness. Equations such as equation 6.2 connect the diagrammatic intuition of a separating hyperplane with a set of movements controlled by parameters such as C , which effectively controls the size of the margin, and α , which effectively bounds the proportion by which a predicted instance can be on the wrong side of the margins that define the hyperplane. In other words, as we have seen previously in cost-function optimisation (see chapter 4), the learning done by the machine consists in finding a way of moving according to these constraints, whose shape has already been inscribed by the functions entering into the

diagram!forms of movement

diagram.

Bending the decision boundary

Substitutions and alterations of the common vector space affect the decision boundary in a second sense. In the abstract of their 1995 paper, Cortes and Vapnik briefly describe the how the support vector machine affects the linear decision surface:

The support-vector network is a new learning machine for two-group classification problems. The machine conceptually implements the following idea: input vectors are non-linearly mapped to a very high-dimension feature space. In this feature space a linear decision surface is constructed (Cortes and Vapnik 1995, 273)

This ‘very high-dimension feature space’ is explicitly made to support ‘a linear decision surface,’ just as Fisher’s linear discriminant analysis had. But this linear decision surface is now the product of a non-linear mapping of the data.¹⁵ In Cortes and Vapnik’s support vector machine, a ‘linear decision surface is constructed’ but in a new domain - ‘a very high dimension feature space’ – where differences exist less linearly. The dimensions of the feature do not index new sources or kinds of data. Instead, the support vector machine creates them from the existing features of the common vector space. From the standpoint of pattern recognition, this sometimes vastly augmented vector space should make it harder to see patterns. In fact, it suggests that not all machine learning

15. The ‘linear decision surface’ is an old and familiar entity in statistics. As we have seen on several occasions, the common vector space invites a certain form of classification based on the search for the best line, the line of best fit, or the most discriminating line, the line that best divides things from each other. Linear regression is not called ‘linear’ for no reason. And Fisher’s ‘discriminant functions’ were later called ‘linear discriminant analysis’ for the same reason: they divide the vector space into different regions (‘decision regions’) separated by ‘linear decision boundaries’ (Alpaydin 2010, 53). Almost all machine learners are aware of and try to address the idealism or abstraction of the line or plane.

reduces the dimensionality of data to diagrammatically flat surfaces. A linear or planar decision surface in high dimensional space maps onto a curving even labyrinthine decision boundary when projected back into the original common vector space. In certain cases, machine learners multiply dimensions in data in the name of differentiation, classification, and prediction. Many of the techniques that have accumulated or been gathered into machine learning do flatten variations and differences into lines and planes, but not always by reducing them. In fact, random forests, neural networks and support vector machines exemplify a counter-movement that maximises variety in the name of differentiation.¹⁶

Research in machine learning, whether it has been primarily statistical, mathematical or computational, countenances and addresses problems of non-linear classification through *dimensional expansion*. As Vapnik writes

in the preface to the second edition of *The Nature of Statistical Learning Theory* (Vapnik 1999, vii), ‘in contrast to classical methods of statistics where in order to control performance one decreases the dimensionality of a feature space, the SVM dramatically increases dimensionality’ (vii).

The powerful variation characteristic of the support vector machine can be understood as a diagrammatic substitution. Consider the expression show

below:

$$L_D = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{i'=1}^N \alpha_i \alpha_{i'} y_i y_{i'} \langle h(x_i), h(x_{i'}) \rangle \quad (6.3)$$

16. Despite the in-principle commitment to any form of function, machine learning strongly prefers forms that can either be visualised on a plane (using the visual grammar of lines, dots, axes, colours, shapes, etc.), or can be computed in form of matrix or vectorised calculations focused on planes. Many of the techniques that grapple with complicated datasets seek to reduce their dimensionality so that lines, planes and regular curves can be applied to them: multi-dimensional scaling (MDS), factor analysis, principal component analysis (PCA), or self-organising maps (SOM) are just a few examples of this.

common vector
space!dimensionality
Vapnik,
Vladimir!dimensional
increase

diagrammatic!substitution

Breiman, Leo!on
support vector
machine

(Hastie, Tibshirani, and Friedman 2009, 423)

In equation 6.3, a re-mapping of the common vector space occurs. All of the data is re-mapped using some function $h(X)$ into a new higher dimensional space. What would be the value of a more complicated space? As Leo Breiman writes in his account of the development of the support

vector machine:

In two-class data, separability by a hyperplane does not often occur. However, let us increase the dimensionality by adding as additional predictor variables all quadratic monomials in the original predictor variables. ... A hyperplane in the original variables plus quadratic monomials in the original variables is a more complex creature. The possibility of separation is greater. If no separation occurs, add cubic monomials as input features. If there are originally 30 predictor variables, then there are about 40,000 features if monomials up to the fourth degree are added.

(Leo Breiman 2001, 209)

The extravagant dimensionality released in the shift from 30 to 40,000 variables vastly expands the number of possible decision surfaces or hyperplanes that might be instituted in the vector space. The support vector machine, however, corrals and manages this massive and sometimes infinite expansion at the same time by only allowing this expansion to occur along particular lines marked out by *kernel functions*. While the support vector machine maintains a commitment to the separating hyperplane, a linear form, albeit with soft margins, it re-constitutes that plane in newly created vector spaces constrained by certain key structural features that render them computationally tractable. On the one hand, a promise of infinite expansion and associated freedom from the rigidity of

lines, and on the other hand, a mode of expansion can only countenance a function!kernel limited range of movements given by the kernel functions.

Hastie and co-authors put it this way:

We can represent the optimization problem and its solution in a special way that only involves the input features via inner products. We do this directly for the transformed feature vectors $h(x_i)$. We then see that for particular choices of h , these inner products can be computed very cheaply (Hastie, Tibshirani, and Friedman 2009, 423)

The terminology here takes us back the common vector space (see Chapter 2) that machine learning inhabits. The ‘inner product’ or ‘the convolution of the dot-product’ described by Cortes and Vapnik come from this space, in which the distances or alignments between whatever can be rendered as

a vector can be calculated en masse. In *Top 10 Algorithms for Data Mining* (Wu et al. 2008), a widely cited computer science account of data mining,

the operation is described in this way:

The kernel trick is another commonly used technique to solve linearly inseparably problems. This issue is to define an appropriate kernel function based on the *inner product* between the given data, as a nonlinear transformation of data from the input space to a feature space with higher (even infinite) dimension in order to make the problems linearly separable. The underlying justification can be found in *Cover’s theorem* on the separability of patterns; that is, a complex pattern classification problem case in a high-dimensional space is *more likely* to be linearly separable than in a low dimensional space (42).

The ‘kernel trick’ that overcomes inseparability remaps an already transformed vector space – the inner product of all the vectors in the data – into a higher dimensional space defined by functions such as $f(x) = x_i^2 + x_i^3$. The trick is no simple technical trick, since as Cortes and Vapnik point out it relies on substantial mathematical developments in the 1960s. ‘The idea of constructing support-vector networks comes from considering general forms of the dot-product in a Hilbert space (Anderson & Bahadur, 1966)’, write Cortes and Vapnik (Cortes and Vapnik 1995, 283). It is a trick, however, in the sense that it is ‘can be computed very cheaply’ as Hastie et. al. put it.¹⁷

What does the transformed feature space combined with the computational short cut of the inner product do in practice? In describing what happens to the generalization error – the errors made when a model classifies hitherto unseen data – Cortes and Vapnik highlight the growth in dimensionality introduced by the technique of the support vector. They describe how the technique exponentially increases the dimensionality of the feature space and how the error rate on difficult to classify handwritten digits drops correspondingly. When the feature space has 256 dimensions (the given dimensions of the 16x16 pixel digits), the error rate is around 12%. As the dimensionality grows to 33,000, then a million, a billion, a trillion and so forth (up to 1×10^{16} dimensions), the error rate drops to just over 4%, which is close to the errors made by ‘human performance’ (2.5%) (288).

¹⁷ This cheapness appeared already in the cat machine learner discussed in the introduction. Heather McArthur’s `skittydar` cat image classifier implemented a support vector machine in Javascript that runs in a browser. Cats are classified cheaply there.

Patterns as institutional not spatial-temporal

The rule of materiality that statements necessarily obey is therefore of the order of the institution rather than of the spatio-temporal localization; it defines possibilities of reinscription and transcription (but also thresholds and limits), rather than limited and perishable individualities (Foucault 1972, 103)

Foucault, Michel!on
statements!materiality
of
statement!materiality of

Patterns in machine learning are a rule of materiality ordering contemporary statements (remembering that for Foucault statements can be machinic: ‘a graph, a growth curve, an age pyramid, a distribution cloud are all statements’ (82)). The spatio-temporality of patterns resides in the ‘order of institution’ rather than the localization because patterns exist in their reiteration and transcription. Pattern recognition is pattern repetition.

Patterns inherently distribute differences. The form of movement engineered into various machine learning techniques grapple with the distribution of differences. But they do it in different ways. Sometimes that take for granted the possibility of separation in a pattern, as if the pattern itself was intrinsically divisible or cleavable along certain lines or contours. At other times, intrinsic inseparability is taken into account as part of the learning. The power of support vector machine to do this is limited, but instructive. It can deal with various forms of inseparability by taking the difficult-to-classify boundary cases as the basis of the model. It deals with problems of non-linearity by increasing the dimensionality of the data, and looking for separations in the higher-dimensional space.

What do we learn about pattern from decision tree and their development into random forests, or from linear discriminant analysis and its re-formalization as the support vector machine? We could also have tracked the movement between the perceptron (Rosenblatt 1958) and the

positivity!of knowledge ‘deep learning’ convolutional neural networks (Hinton and Salakhutdinov 2006) of more recent practice (see chapter 8 . Each of these shifts attests, I have been suggesting, to the emergence of a new enunciative mode in which dispersion and regularity, accumulation and rarity are no longer opposed, or treated as contradictory. A single decision tree becomes thousands in a random forest. A relatively small number of dimensions in the common vector space becomes potentially infinite in the convolutional dot products and kernel functions of support vector machines. Models that sought to encompass or fit everything in the data including the outliers within a single probability distribution instead dwell on the difficult-to-classify, the erroneous or borderline instances amidst the massive normalized accumulations of event.

What counts as pattern today? Recent machine learners can be studied, I have been suggesting as elements in a power-laden positivity, that changes knowledge in specific, actually quite narrow, but nevertheless far-reaching ways. The interpretability of a decision tree cascades into the statistical regularities associated with a random forest of many thousands of trees. The separating lines and planes that allow linear models to become classifiers in the ‘classic’ techniques such as linear discriminant analysis find themselves displaced into hyper-planes, into newly constructed and sometimes inordinately-dimensioned feature spaces that can only be traversed by virtue of the kernel functions, and their computationally tractable inner products. These transformations are not necessarily scientific, although various sciences make great use of them, but they do diagram thresholds of epistemologization and formalization in practice. Despite their quite different ways of partitioning, separating or propagating differences, support vector machines and decision trees define possibilities of grouping and spacing, sometime through purifying, sometimes through bending and blurring.

These groupings and spacings attract, generate and accumulate propositions. This grouping-spacing positivity, space, despite its commonalities, is not an homogeneous field. It does not have the coherence of a science, it uses different systems of formalization (the cross-entropy measures of the decision tree, the gradient descent of the neural network, the dual form Lagrange multipliers of the support vector machine), and disperses in different ways across knowledge practice (the decision tree with its commercial uptake in data mining versus the support vector machine's heavy use in image recognition and classification). As we will see in the following two chapters, deep ordering differences such as living and non-living, saying and seeing, making and using might begin to take on new patterns.

Title	Year	twidhC- i- ta- tions
A tutorial on Support Vector Machines for pattern recognition	1998	twidh3510
A tutorial on Support Vector Machines for pattern recognition	1998	twidh3481
Support vector machine classification and validation of cancer tissue samples using microarray expression data	2000	twidh869
Choosing multiple parameters for support vector machines	2002	twidh681
Classification of hyperspectral remote sensing images with support vector machines	2004	twidh428
Comparing support vector machines with Gaussian kernels to radial basis function classifiers	1997	twidh405
Support Vector Machines for 3D object recognition	1998	twidh396
Comparing support vector machines with Gaussian kernels to radial basis function classifiers	1997	twidh392
Comparing support vector machines with Gaussian kernels to radial basis function classifiers	1997	twidh353
An assessment of support vector machines for land cover classification	2002	twidh295
An assessment of support vector machines for land cover classification	2002	twidh283
Drug design by machine learning: support vector machines for pharmaceutical data analysis	2001	twidh273
Drug design by machine learning: support vector machines for pharmaceutical data analysis	2001	twidh268
A novel method of protein secondary structure prediction with high segment overlap measure: Support vector machine approach	2001	twidh266

control!crisis of
data!DNA microarray

Chapter 7

Regularizing objects

The opening pages of *Elements of Statistical Learning* present four vignettes – spam email classification, handwritten digit recognition, prostate cancer risk and ‘DNA Expression Microarrays’ – and list five examples (document classification, image recognition, risk of heart attack, stock price prediction, risk factors for prostate cancer, and glucose estimates for diabetics) (Hastie, Tibshirani, and Friedman 2009, 1-7). What happens to things like prostate cancer, handwritten digits or stock prices when machine learners traverse them? The encounter between machine learning and things in the world takes the form of a control crisis (Beniger 1986), a situation in which the viscous temporality and inter-objectively distributed properties of the things, occasions new referentialities and positivities of knowledge and practice.

The last of the vignettes attracts a whole page colour figure – a heatmap – of microarray data (Hastie, Tibshirani, and Friedman 2009, 7)¹. DNA, genes, genomics and proteomics then more or less disappear from view for the next 503 hundred pages of the book (aside from a brief mention in the context of cross-validation), only to reappear somewhat suddenly in a discussion

1. I have discussed the history of heatmaps and their place in contemporary science in other work (Mackenzie 2013a).

machine
learner!hierarchical clustering
genomics!importance in machine learning
referential!entanglement
data!strain

of unsupervised machine learning techniques (k-means, agglomerative and hierarchical clustering; Chapter 14, where the DNA microarray data is re-analysed using hierarchical clustering), and then again, and much more resoundingly, in a final chapter (Chapter 18) new to the second edition of the book on ‘High Dimensional Problems.’ Apart from one example where the Hastie and co-authors develop a document classifier for their journal articles, every example in the added Chapter 18 comes from genomic science, a scientific field that largely begins to take a recognisable shape in the late 1990s as both sequence data and high-throughput DNA-analysis devices, particularly microarrays, become widely available. Alongside spam filtering, image or handwriting recognition, genomic research into biological processes has a strongly referential effect on machine learning. It ‘forms the place, the condition, the field of emergence, the authority to differentiate between individuals or objects, states of things and relations that are brought into play by the statement itself; it defines the possibilities of appearance and delimitation of that which gives meaning to the sentence, a value as truth to the proposition’ (Foucault 1992 [1966], 91).

This referential entanglement happens in several different ways. The first concerns what I earlier called data strain. Genome data volume inflates the common vector space. Genomics generates new versions of the now familiar problems of data dimensionality. It produces data whose abundance, diffusion, heterogeneity or impaction thwarts its examination, tabulation, and regulated circulation. Sometimes genomics data exhibits unusual mixtures of accumulation and sparsity. Clinical genomics in particular generates datasets that are lavishly furnished with ‘features’ but often quite meagrely supplied with clinical cases or ‘observations’. In the shorthand typical of machine learning terminology, p is larger than N : ‘the number of features p is much larger than the number of observations N , often written $p \gg N$ ’

(Hastie, Tibshirani, and Friedman 2009, 649). This strains statistical methods that rely on the ‘Law of Large Numbers’ (Hacking 1990, 99-104), which holds that the accuracy of statistics tends to increase with more observations.

statistics!
Law of Large
Numbers
positivity
function!
biological

The second referential entanglement comes from the *positivity* of genomes themselves. Since the early nineteenth century, biology and cognate disciplines have sought to know and understand problems of time, genesis, duration, activity and process in a very broad spectrum of living things. But contemporary genomics seeks to do so, as many commentators have noted, by concentrating and coalescing different operational, algorithmic and mathematical senses of function around the long DNA sequences comprising genomes (see chapter 4). . The primary ‘object’ in genomics is a genome, the full complement of DNA in an organism. Genomes vary in size from the 2000 DNA base pairs of a virus, the 3.2 Gb (gigabase pairs) of humans through to the 130 Gb of the lung fish. The founding premise of genomic science is that knowing the complete sequence of DNA potentially yields insight into biological processes of many different kinds, ranging from evolution (phylogeny), development (ontogeny), metabolism, structure and pathology. If nothing else, genome comparison promises insights into 3.8 billion years of evolution. In all of these respects, DNA sequences have since at least the 1980s served as the common substrate for many different scientific experiments, technical developments, cyber-infrastructures and needless to say, biological imaginaries oriented around the problems of control.² The genomic premise has an ineluctably promissory aspect since prior to the whole genome sequencing projects initiated in the 1990s, biologists had never worked with genomes only with selected DNA sequences, especially

2. A large and very diverse social science and humanities literature now exists around genomics. I draw on some of that literature as general background here, especially (Sunder Rajan 2006; Thacker 2005; Stevens 2011; Leonelli 2014) and (Haraway 1997), but largely do not address it directly.

those associate with genes and the proteins that they code. The genome, with all the many duplicated, redundant, and slightly varying patterns of DNA, bears the traces of long evolutionary in-mixation and constitutes a massive functional process whose exquisite sensitivity to changing conditions – a slight change in light reaching a leaf cascades can be traced in genomic functions – forms a limit case for any operational sense of function. The functioning of genomes symbolises a deeply interconnected relationality in life sciences, and becomes the test case for the learning capacities of machine learners.³

As referential field for machine learning, genomes pose a problem of unregulated referentiality. DNA sequences exist in great abundance (in databases, and increasingly, from the cheaper and more compact sequencing technologies), yet even determining how DNA sequence fragments can be ordered in the genome – let alone how they make sense as some biological function – is much harder. They are assembled as datasets via statistical models.

3. As data forms, genomes have a problematic mode of existence. They resemble cat images on the internet. As a data form, genomes are remarkably homogeneous. They are one-dimensional strings of letters corresponding to the well-known four nucleic acids (g, a, t, c). While many earlier tabulations of variation, difference, groups, types and relations are woven through the life sciences, genomes have for the last several decades mesmerised biological sciences as a way of analysing and re-distributing the confused multiplicities associated with living things. The raw data for genomes comes from the sequencing of DNA obtained from various organisms - viruses, bacteria, plants, fish, animals and humans. The sequencing of DNA, especially DNA that encodes the proteins that pervade biological processes, that structure tissues or assemble in complicated metabolic pathways, has been the concern first of molecular biology (mainly in the 1970s-1990s) and more recently genomics (post-1990). In molecular biology, DNA sequences were carefully elicited (using the experimental techniques for instance of Sanger sequencing) and then compared with already known sequences of DNA to identify similarities that might have biology significance (for instance, evolution from a common ancestor). In genomics, DNA sequences generally originate from increasingly high-throughput sequencers that output massive datasets (see (Stevens 2013; Mackenzie et al. 2015)). Given both the accumulated store of already sequenced DNA and the increasingly viable practices of sequencing all of the DNA in a given organism, genomics has promised a much wider and more detailed understanding of biological complexity than any previous life science had been able to obtain. With genomes in hand, biologists for the first time would be in a position to build models of entire domains of biology, domains that previously could only be explored through painstaking experiments targeting specific cells, molecules, biochemical reactions and networks. The vast yet somewhat dispersed knowledges of the life sciences might be re-ordered and aligned on a new very extensive yet quite homogeneous backbone of the genome read out as billions of DNA base pairs.

'Genome assembly continues to be one of the central problems of bioinformatics' write the authors of a recent scientific review of the techniques of constructing whole genomes from DNA sequencer data (Henson, Tischler, and Ning 2012). Even the elementary data form of the genome as DNA base pairs is a highly algorithmic construct. No existing sequencing technology produces a genome as a single sequence, as a vector (in the sense described in chapter 3). Instead, sequencing produces randomly sets of sequence fragments of various lengths that have to be assembled into a complete genome algorithmically. Whole genome assembly as reported for the initial draft of the human genome in 2001 (Venter et al. 2001; Lander et al. 2001) or for the model biological organism, *Drosophila* (Myers et al. 2000) were not at the time understood as machine learning tasks. But the problem of whole genome assembly from DNA fragments was seen as probabilistic in the sense that the aim is to assemble the often millions of short sequence fragments in an order that is most likely to occur. Even prior to the first full human genome assembly, genomic science had made heavy use of probabilistic models in aligning DNA (and protein amino acid) sequences.⁴ The practical problem here is that genomes contain swathes of duplicated regions that make assembling sequences in good order a severe challenge. While sequence alignment algorithms have long used algorithmic approaches (known as dynamic programming) to score the similarity between two given DNA sequences, assembling the millions of DNA sequences produced by

4. Richard Durbin, Sean Eddy, Anders Krogh and Graeme Mitchison's highly cited *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids* (Durbin et al. 1998) was based almost entirely on Hidden Markov Models, a way of modelling a sequence of states that *Elements of Statistical Learning* treats at chapter length (see (Hastie, Tibshirani, and Friedman 2009, Chapter 17)). While sequence alignment was regarded as a deeply algorithmic and statistical problem in the former volume, it is not at all formulated in the language of machine learning. There is little discussion of cost functions, vector spaces, optimisation, problems of generalization, supervised or unsupervised learning. On the other, David Haussler, a key bioinformatician in the first draft of the human genome in his work explicitly sought to bring machine learning methods to bear on biology, and continues to do so. See (Zerbino, Paten, and Haussler 2012) for a review of the relevance of machine learning to genomic science.

hyperobject!genome as contemporary sequencers has necessitated entirely new techniques (shifting, for instance, from the overlap-layout-consensus model to the de Bruijn graph-based path models (Pevzner, Tang, and Waterman 2001).

Despite the intensive work on genome assembly [see (Henson, Tischler, and Ning 2012) for review), all genome assemblers produce errors. Put differently, whatever else might be known on the basis of the genome (genes, mutations, evolutionary relationships, variations associated with disease, heredity or individual identity), the genomic data and hence the genome itself as a scientific hyperobject is deeply probabilistic. The assemblers - the algorithmic process producing whole genomes from many fragments – order sequences according to the same statistical criteria of maximum likelihood used in machine learning models. They rely heavily, moreover, on existing biological databases for reference or anchor points. No matter what assembly algorithm is used, the dependencies on other datasets and other knowledges, the perseverance of errors, and the deeply probabilistic texture of what counts as sequence data remain as key features of genomes. The indelible errors, the entangled dependences on accumulated biological knowledges and above all the deeply probabilistic rendering of DNA as biological bedrock make genomes a particularly challenging site of machine learning activity.

Hastie and co-author's invocation of DNA-related data, therefore, is no arbitrarily chosen example amidst the general proliferation of settings, domains, cases and examples typically found in machine learning pedagogy. In multiple dimensions and directions, genomics – the scientific project of operating on the whole DNA complement of organisms – is a nearest neighbour, a tightly adjacent territory of machine learning even if relatively few machine learners have, to date, managed to work with whole genome sequence data. This relatively long-established proximity (at least 25 years, and perhaps more) of genomics and DNA-based data is strategi-

cally important in the diagrammatic generalization or diagrammatization of machine learning, in the processes whereby the diagrammatic forms configured around these techniques, with their specific forms of articulation, statement and making-visible, propagate into multiple, once-disparate settings.⁵ Like social media platforms or retail spaces with their many visitors, genomes, I would suggest, provoke a multiplicity of machine learners to bind to them like antibodies to an antigen (or an allergen). Genomes function as regularizing hyperobjects for machine learning. At the same time, machine learners profoundly re-configure the economically charged fields of biomedicine and biological research and generate new modes of action, accounting, testing, and examining biological processes (metabolism, development, disease), relations (species, evolutionary, commensual), and functions (reproduction, digestion, photosynthesis, immunity, circulation, etc.). Wide-ranging infrastructural, institutional, professional and financial arrangements follow along new paths constructed by machine learning, with its epistological transformations, its function-finding, its regimes of probabilistic, decisionistic and re-patterned engagement with data.

diagram!diagrammatization!
as
generalization
machine
learning!regularizing
hyperobjects

The positivity of the genome

```
## Error in table(actual_topics): attempt to set an attribute on NULL
```

In research done on machine learning during 1990-2015, biology, and particularly molecular and then genomic biology, has a very high profile in the

5. Signal processing is another such domain. Many of the techniques now prominent in machine learning developed in parallel in signal processing, where the encoding and decoding of signals has long been seen as a problem of pattern recognition amenable to statistical calculation. In some specific cases, such as Hidden Markov Models, the same techniques seem to appear almost simultaneously in very disparate domains. Hidden Markov Models appear in genomics (as part of the problem of sequence alignment) at the same time as they begin to appear in digital signal processing for wireless communication and video image compression (Mackenzie 2010) and above all, in speech recognition (Rabiner 1989).

hyperobject
 genome!as hyperobject
 genomics!as
 cross-validation of
 machine learning
 referential!epistemolo-
 gization! threshold
 of
 vectorization!infrastructural
 Google!I/O Conference,
 2012

research publications. (See table ??.) After the leading machine learning disciplines (computer science, electronic engineering and statistics), molecular biology, genomics and bioinformatics attract most academic journal citations and publications associated with machine learning. Half of the most cited research literature has a biomedical or life science referentiality. This may be because genomes and human disease in particular, are premiere scientific hyperobjects like the human brain, dark matter, global climate or fundamental particles in contemporary sciences. But it might also be the case – and I will pursue this line of argument here – that genomes, with all their operational and functional complexity, takes shape in concert with machine learning. In terms of contemporary biological knowledge production, the transformation of biology into a data-intensive science (Hey, Tansley, and Tolle 2009; McNally et al. 2012) is tightly entangled with machine learning in processes of cross-validation.

In the generalization of machine learning, genomics and the genome referential mark a threshold of epistemologization. Genomes are both a challenge to the capacity of machine learning to produce scientific knowledge (as distinct from say the unstable commercial knowledge of a credit risk model), and a way of validating machine learning as a life-death relevant knowledge practice. Genomes also provide the pretext for vectorizing infrastructural re-configurations such as computational clusters, grids, arrays and clouds. For instance, the Google Compute Engine, a globally addressable ensemble of computers typical of recent distributed commercial computing architectures, was briefly turned over to exploration of cancer genomics during 2012, and publicly demonstrated at the annual Google I/O conference. Midway through the demonstration, in which a human genome is visualized as a ring in ‘Circos’ form (see figure ??fig:circos} (Krzywinski et al. 2009)), the speaker, Urs Hözle, Senior Vice President of Infrastructure at Google

'then went even further and scaled the application to run on 600,000 cores across Google's global data centers' (Inc. 2012). The audience clapped as the annular diagram of a human genome was decorated with a rapidly increasing number of cross-links, accompanied by a snapping sound as it appeared. The world's '3rd largest supercomputer', as it was called by TechCrunch, a prominent technology blog, 'learns associations between genomic features' (Anthony 2012). Note the language of machine learning: it 'learns ... associations between features.' We are in the midst of many such demonstrations of 'scaling applications' of data in the pursuit of associations between 'features.'⁶

The I/O conference audience, largely comprising software developers, could hardly be expected to have a detailed interest in what was being shown on the screen. Their interest was steered toward the immediate availability of computing power: from 10,000 to 600,000 cores in a few seconds. Such drastic infrastructural re-scaling attests to the provocative form of the genomic referential. The principal chain of associations validated by the genomics demonstration was, presumably, something more indexical: genome=>complexity=>cancer/disease=>life/death=>important. Yet the Google Compute demonstration is, I would suggest, typical of how genomes, genes, proteins and biological sciences more generally, have supply positivity to machine learning . This positivity is only hinted at in the Google I/O keynote address in Hölzle's talk of genomic features, gene expression and patient attributes. The only concrete indication of how what was happening in the demonstration related to machine learning was one mention of the RF-ACE (Random Forest- Artificial Contrasts with Ensembles) algorithm.

6. A second significant and equally prestigious example of this infrastructural re-scaling might be IBM Corporation's 'cognitive computing platform,' Watson. Watson, a distributed computing platform centred on machine learning, is hard to delineate or easily describe since it exists in a seemingly highly variable form. Its uses in genomics, pharmaceutical discovery, clinical trials and cooking are heavily promoted by IBM (IBM 2014). Another would be Amazon Web Services various cloud computing services, some of which have been heavily used by genomic scientists.

Google!Google Compute Engine
graphic!Circo diagram
positivity

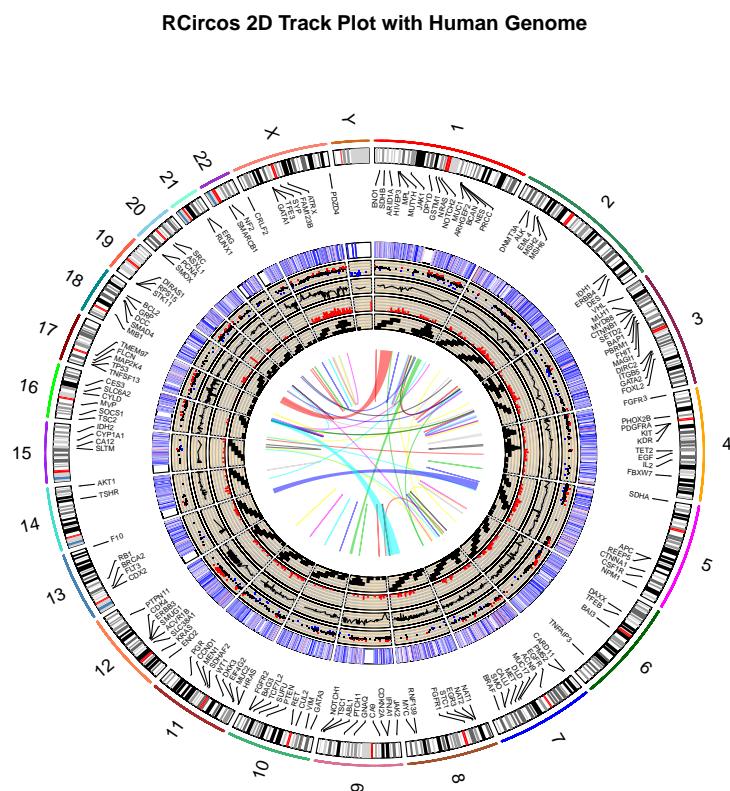


Figure 7.1: A human genome diagrammed using the Circos form. The many tracks of this diagram support a range of graphic forms including scatterplots, heatmaps and histograms all anchored to the ideogram of the 23 chromosomes of the human genome.

Google's press release emphasises epistemic values:

machine
learner!RF-ACE

it allows researchers to visually explore associations between factors such as gene expression, patient attributes, and mutations - a tool that will ultimately help find better ways to cure cancer. The primary computation that Google Compute Engine cluster performs is the RF-ACE code, a sophisticated machine learning algorithm which learns associations between genomic features, using an input matrix provided by ISB (Institute for Systems Biology). When running on the 10,000 cores on Google Compute Engine, a single set of association can be computed in seconds rather than ten minutes, the time it takes when running on ISB's own cluster of nearly 1000 cores. The entire computation can be completed in an hour, as opposed to 15 hours (Inc. [2012](#)).

Amidst this mire of fairly technical computing jargon, we might observe that Google applies an algorithm developed by engineers at Intel Corporation and Amazon to genomic datasets provided by the Institute of Systems Biology, Seattle, a doyen of big-data genomics. The RF-ACE algorithm (a further development of Breiman's random forests discussed in chapter 6) literally re-draws a diagram of the genome, and re-draws it increasingly rapidly as the demonstration scales up to 10,000 cores (or CPUs). A diagram that normally appears statically on-screen or on the printed page of a scientific publication is now animated by an algorithmic process. This confluence of commerce (Amazon), industry (Intel), media (Google) and genomic science (ISB) exemplifies the diagrammatic generalization of machine learning.

In none of these demonstrations and examples, whether they come from *Elements of Statistical Learning* or from Google Compute Engine, does the object of the knowledge – genomes, genes, proteins – need to figure in terms

positivity!as basis of propositions

of its original discipline or scientific field (typically cancer biology). The de-coupled ‘positivity’ of the genome, to use Michel Foucault’s term from *The Archaeology of Knowledge*, functions as a matrix from which propositions are generated.

As Foucault writes:

To analyse positivities is to show in accordance with which rules a discursive practice may form groups of objects, enunciations, concepts, or theoretical choices. The elements thus formed do not constitute a science, with a defined structure of ideality; their system of relations is certainly less strict; but neither are they items of knowledge piled up one on top of another, derived from heterogeneous experiments, traditions, or discoveries, and linked only by the identity of the subject that possesses them. They are that on the basis of which coherent (or incoherent) propositions are built up, more or less exact descriptions developed, verifications carried out, theories deployed. They form the precondition of what is later revealed and which later functions as an item of knowledge or an illusion, an accepted truth or an exposed error, a definitive acquisition or an obstacle surmounted (Foucault 1972, 181-2)

In the case, the positivity of RF-ACE and its scaled-up demonstration on Google Compute consists in the system of relations it groupwise assembles between cancer patients, vectorised infrastructures and predictive classifications. Similarly, the treatment of DNA microarray data in the slightly earlier examples found in *Elements of Statistical Learning* does not principally concern cancer biology as such, but much more the way a group of elements are assembled so as to permit the production of propositions that cross the threshold of scientificity if configured in relation to experimental

practices and scientific literatures, but may just as well posit different forms of governmental, market-focused, organisational or managerial operations.⁷

The advent of ‘wide, dirty and mixed’ data

The plurality of applications can sometimes make it seem that machine learning docks in the different domains, and then proceeds to administer learning algorithms to existing knowledges practices wherever it finds them. The law of generalization here would seem to be an epistemic *terra nullius* doctrine, in which existing knowledge titles are rapidly extinguished by algorithmic processes. We have seen previously that ancestral communities of probabilisation orient the generalization of machine learning (see chapter 5). The research literature published on machine learning since the early 1990s clusters around several main problems – image recognition, document classification, and segmenting market behaviour (as in, working out what advertisement to show, or whether someone is likely to buy a particular product, etc.). These problems position machine learning amidst regimes of visibility, the production of economic value, and the regularities of statements (or put in more Foucaultean terms, amidst life, labour and language; see (Foucault 1992 [1966])). Where, amidst these major regularities, does molecular biology or genomics (arguably the successor of molecular biology) fit? Almost all of the major machine learners, albeit supervised or

7. In their account of the surprisingly slow shift of microarrays towards clinical practice, Paul Keating and Alberto Cambrosio identify statistics as a kind of bottleneck:

The handling and processing of the massive data generated by microarrays has made bioinformatics a must, but has not exempted the domain from becoming answerable to statistical requirements. The centrality of statistical analysis emerged diachronically, as the field moved into the clinical domain, and is re-specified synchronically depending on the kind of experiments one carries out (Keating and Cambrosio 2012, 49).

What Keating and Cambrosio describe as ‘becoming answer to statistical requirements’ I would suggest also entails a transformation of statistical requirements in a new operational diagram that reduces some of the frictions associated with existing statistical practice. This operational diagram is machine learning.

cross-validation!referentiality of

unsupervised, discriminative or generative, parametric or non-parametric, substantial research activity during the last two or so decades cross-validate their statements with genomics. (The difference between the first and second editions of *Elements of Statistical Learning* attests to that tendency.)

We can see this cross-validation at work in the treatment of genomic data in *Elements of Statistical Learning*. As to the former problem of dimensionality, both the RF-ACE cancer biology demonstration and the DNA microarray data extensively modelled in the final chapter of *Elements of Statistical Learning* address some elementary problems associated with genomic data. If we turn back to the `iris` dataset (R. A. Fisher 1936), perhaps the most heavily used pedagogical dataset in the literature, it is strikingly obvious that the data does not provoke the infrastructural contortions associated with Google Compute, or for that matter, the highly sophisticated and quite subtle treatment of gene expression we find in genomics-related machine learning.

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.10	3.50	1.40	0.20	setosa
4.90	3.00	1.40	0.20	setosa
4.70	3.20	1.30	0.20	setosa
4.60	3.10	1.50	0.20	setosa
5.00	3.60	1.40	0.20	setosa

Table 7.1: First 5 rows of Fisher’s ‘iris’ dataset

It is usual, in working with `iris`, to construct machine learners that use the variables from the first four columns shown in ?? to infer the value of the `Species` variable (as seen in Chapter 5, where a decision tree was constructed using this same dataset). The measurements of petals and sepals of the irises of the Gaspé Peninsula in Novia Scotia, and their

classification into different species is perhaps a typical mid-twentieth century dataset!iris biological procedure. The technique of linear discriminant analysis that data!wide, dirty, mixed Fisher demonstrated as a way of classifying the species of iris continues in use, but the shape and texture of contemporary datasets differs greatly from what we see in this table. Even in the excerpt shown in Table ??, we can see that it is quite narrow as it has only a few columns, the data is nearly all of one type (measurements of lengths and widths), and the data is clean (there are no missing values). The data is tightly contained in the table. Iris is typical of classic statistics, and much biological data prior to genomics in its relatively homogeneity and distinct partitioning.

If `iris` is the conventional form, how does a genomic dataset differ? One clue comes from descriptions of the RF-ACE algorithm, first published in 2009. RF_ACE is attempts to deal with ‘modern data sets’ that are ‘wide, dirty, mixed with both numerical and categorical predictors, and may contain interactive effects that require complex models’ (Tuv et al. 2009, 1341). Such algorithms and the ‘wide, dirty, mixed’ datasets they work on have a distinctive texture, which, I would suggest, we should try to grasp if we want to understand how genomic data has become a lightning rod for machine learning. More clues come from the various treatments of DNA microarray data in *Elements of Statistical Learning*. Hastie and co-authors introduce one microarray dataset they use in this way:

The data in our next example form a matrix of 2308 genes (columns) and 63 samples (rows), from a set of microarray experiments. Each expression value is a log-ratio $\log(R/G)$. R is the amount of gene-specific RNA in the target sample that hybridizes to a particular (gene-specific) spot on the microarray, and G is the corresponding amount of RNA from a reference sample. The samples arose from small, round blue-cell tumors

dataset!SRBCT
 science!knowledge!positivity
 of

(SRBCT) found in children, and are classified into four major types: BL (Burkitt lymphoma), EWS (Ewing's sarcoma), NB (neuroblastoma), and RMS (rhabdomyosarcoma). There is an additional test data set of 20 observations. We will not go into the scientific background here (Hastie, Tibshirani, and Friedman 2009, 651)

Note that while the number of samples (~80) in the small round blue-cell tumors (SRBT) (Khan et al. 2001) dataset is less than the number of flowers measured in `iris`, the number of variables presented by the columns in the table (2308) is much greater. Hastie and co-authors, like the Google I/O demonstration, do not ‘go into the scientific background.’ Scientific knowledge *per se* is not the central concern in machine learning. Rather, but the positivity of knowledge is important. The original publication of this dataset in 2001 (Khan et al. 2001) also made use of machine learning techniques (neural networks, a major topic in the next chapter 8), but precisely in order to address the diagnostic difficulties of distinguishing different types of such tumors without resort to new experiments or biological knowledge.⁸

The sample of the SRBCT data shown in Table ?? does not readily accommodate the wide table of this dataset. Unlike `iris`, the thousands of variables simply cannot be displayed on a page or screen. *Wide* datasets are quite common in machine learning settings generally, but particularly common in genomics where in a given study there might only be a relatively small

8. Khan and co-authors write:

Gene-expression profiling using cDNA microarrays permits a simultaneous analysis of multiple markers, and has been used to categorize cancers into subgroups 5–8. However, despite the many statistical techniques to analyze gene-expression data, none so far has been rigorously tested for their ability to accurately distinguish cancers belonging to several diagnostic categories (Khan et al. 2001, 673)

GENE1	GENE2	GENE3	GENE4	GENE5	GENE6	GENE7	GENE8	GENE9	GENE10	GENE11
0.77	-2.44	-0.48	-2.72	-1.22	0.83	1.34	0.06	0.13	0.57	0.00
-0.08	-2.42	0.41	-2.83	-0.63	0.05	1.43	-0.12	0.46	0.16	0.00
-0.08	-1.65	-0.24	-2.88	-0.89	-0.03	1.16	0.02	0.19	0.50	0.00
0.97	-2.38	0.63	-1.74	-0.85	0.95	1.09	0.82	-0.28	0.99	0.00
0.08	-1.73	0.85	0.27	-1.84	0.33	1.25	0.77	0.03	0.28	0.00

Table 7.2: Small round blue-cell tumour data sample (Khan, 2001)

number of biological samples but a huge amount of sequencer or microarray data for each sample. This wide dimensionality of the data is common. Note too that while *Elements of Statistical Learning* picks up the SRBCT dataset from biomedical research ((Khan et al. 2001) appears in the journal *Nature Medicine*), many other similar datasets appear in the machine learning literature. Much genomic data shares this generic feature of width.⁹

```
## Error in eval(expr, envir, enclos): object 'train' not found
```

9. Importantly, as discussed in Chapter 2 (in terms of the diagonalization running between different elements of code, data, mathematical functions and indexical signs) and in Chapter 3 (in terms of the auratic power of datasets), the fact that these datasets can be so readily loaded and accessed via bioinformatic infrastructures using code written in R or Python is also a notable feature of their advent in the machine learning literature. Even a social science researcher can quickly write programs to retrieve this data. It attests to several decades, if not longer, work on databases, web and network infrastructures, and analytical software, all, almost without exception, driven by the desire for aggregation, integration, archiving and annotation of sequence data that first became highly visible in the Human Genome Project of the 1990s. The brevity of these lines of code that retrieve and load datasets – half a dozen statements in R, no more – suggests we are dealing with a high-sedimented set of practices, not something that has to be laboriously articulated, configured or artificed. Code brevity almost always signposts highly-trafficked routes in contemporary network cultures. Without describing in any great detail the topography of databases, protocols and standards woven by and weaving through bioinformatics, the ready invocation of genomic datasets suggests that the mixed, dirty, wide datasets fed to algorithms such as RF-ACE or analysed in (Hastie, Tibshirani, and Friedman 2009) derives from the layered couplings and interweaving of scientific publications and scientific databases developed by biological science over the last three decades. As the code shows, sequence and other genomic data are available to scientists not only as users searching for something in particular and retrieving specific data, but to scientists as programmers developing ways of connecting up, gathering and integrating many different data points into to produce the wide (many-columned), mixed (different types of data), and dirty (missing data, data that is ‘noisy’) datasets, datasets whose heterogeneous and often awkward topography then elicits and invites algorithmic treatment.

datasets!diagrammatic character of

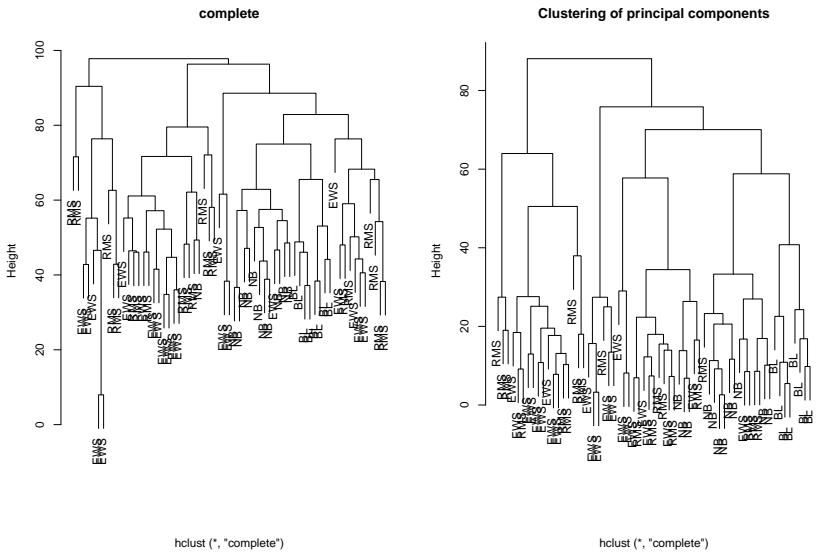


Figure 7.2: Hierarchical clustering of the SBRCT gene expression data

```
## Error in NextMethod("[") : object 'train' not found
```

In contrast to the `iris` data, the SRBCT data becomes more highly diagrammatic in the sense that it both occludes and diagonally connects many levels of practice. The columns in table ?? refer to genes whose levels of expression in different samples are measured by comparison to their levels in a reference sample. Similarly, the hierarchical clusterings of the data shown in figure 7.2 The very identification of the several thousand genes whose levels of expression are measured by the microarray experiments presupposes much preceding work on DNA sequences and their identification as protein-coding DNA amidst the highly repetitive vector of the genome sequence. Highly leveraged infrastructures for access to biological data underpin such datasets. Considered more diagrammatically, genomes in many ways becomes less linear or flat than the base sequences might suggest. (Even these sequences themselves harbour, as we will see, probabilistic models). The linear sequences of DNA data becomes more mixed and wide partly through the accessibility we have just seen that allows them to be superimposed, annotated and layered. A recent review in the

journal *Genomics* highlights the increasing importance of machine learning techniques:

High-throughput genomic technologies, including gene expression microarray, single Nucleotideide polymorphism (SNP) array, microRNA array, RNA-seq, ChIP-seq, and whole genome sequencing, are powerful tools that have dramatically changed the landscape of biological research. At the same time, large-scale genomic data present significant challenges for statistical and bioinformatic data analysis as the high dimensionality of genomic features makes the classical regression framework no longer feasible. As well, the highly correlated structure of genomic data violates the independent assumption required by standard statistical models(Chen and Ishwaran [2012](#), 323).

This kind of commentary on the changing shape, not just the volume, of genomic data is quite common. Such contrasts typically highlight the incompatibility between a surging multiplicity of data forms and the constraints of existing statistical modelling techniques ('standard statistical models'). Newer instruments or devices such as microarrays and faster sequencers (so-called 'next generation sequencers') loom large (Mackenzie [2015a](#)). The tropes of waves, deluges, floods and waves of DNA sequence and microarray data being somewhat washed out, this account instead highlights the 'high dimensionality of genomic features' and the 'highly correlated structures of genomic data.'

Regularizing machine learning in genomics

Machine learners working with sequence data typically highlight changes in statistical approaches. So for instance, Chen and co-authors recommend

machine learner!random forest!use in genomics the use of the random forest (RF) algorithm because it:

is highly data adaptive, applies to “large p, small n” problems, and is able to account for correlation as well as interactions among features. This makes RF particularly appealing for high-dimensional genomic data analysis. In this article, we systematically review the applications and recent progresses of RF for genomic data, including prediction and classification, variable selection, pathway analysis, genetic association and epistasis detection, and unsupervised learning (Chen and Ishwaran 2012, 323)

Familiar machine learning keywords such as ‘large p, small n’, ‘high dimensional’, ‘prediction’, ‘classification,’ ‘variable selection’ and ‘unsupervised learning’ pepper their recommendations. While these terms are widely used in machine-learning research since the early 1990s, they are becoming increasingly visible in genomics. The key terms on the genomics side of this formulation would perhaps be ‘pathway analysis’, ‘genetic association,’ and ‘epistasis.’ These biological terms point to forms of relationality typically associated with biologically interesting processes. Epistasis for instance broadly refers to linked gene action, a process that has been difficult to study before high-throughput methods of functional genomics were developed. In contemporary genomic science, these biological processes are increasingly understood in terms of eliciting and modelling the relations between *features* of genomic datasets in order to classify and predict biological outcomes. In between the machine learning and the genomic references appear several statistical terms: ‘correlation’ and ‘interaction.’ How does machine learning differ from the statistical practice that has underpinned much of modern biology?

The change concerns a mutual articulation between genomic and machine learning research. The analysis of SRBCT gene expression in *Elements of Statistical Learning* (as introduced above) is symptomatic of this mutual articulation, a cross-validation that epistemologizes both genomics and machine learning. On the one hand, the genomics promises a read out of all the genes in a given organism (~20,000 for humans). On the hand, the pattern of activity of these genes in time, or any particular point in the life of an organism, cannot be read from the genome but only in time-varying expression, the changes in state and the variations in closely similar genomes. The overt arrival of machine learning techniques in genomic research was initially largely concerned with this problem of gene expression in time (and in fact, nearly all of the analysis of genomic data in *Elements of Statistical Learning* explicitly deals with various cases of gene expression).

Compared to the algorithmic craft seen in whole genome assembly (Venter et al. 2001; Myers et al. 2000; Pevzner, Tang, and Waterman 2001), the handling of DNA-based data in machine learning settings can seem rather crudely lacking in biological specificity. Hastie and co-authors almost deprecate scientific knowledge: ‘we will not go into the scientific background here.’ Like the authors of the original scientific study (Khan et al. 2001), *Elements of Statistical Learning* treats gene expression profiling largely as a problem of classification. The many gene expression studies seek to discriminate between different conditions, diseases, or pathologies on the basis of differing levels of gene expression. For machine learners, each gene is a variable whose levels of expression in a given sample may help identify what type that sample belongs to. In the case of the SRBCT data, the types include lymphomas, sarcomas and neuroblastomas.

The shape of the data, a shape that owes much to the great accumulation of knowledge associated with molecular biology, immediately poses problems.

machine learner!linear discriminant analysis!not applied to gene expression machine learning!regularization in Foucault, Michel!disciplinary power machine learning!regularization in

This shape, as well will see, only becomes more problematic in the later proliferation of sequence data associated with whole genome sequencing. ‘Since $p \gg N$ ’ write Hastie and co-authors, ‘we cannot fit a full linear discriminant analysis (LDA) to the data; some sort of regularization is needed’ (Hastie, Tibshirani, and Friedman 2009, 651). Why cannot standard machine learners such as linear discriminant analysis fit here? What happens when machine learners encounter wide genomic datasets? ‘Some sort of regularization’ is needed they add.

What is this regularization? Like the re-distribution of randomness into a population of machine learners (see chapter 5, regularization governs an potentially unruly plurality through a form of observation. Michel Foucault describes the advent of disciplinary power partly in terms of enclosure or individualizing observation, but also in terms of techniques of supervising, examining and above all, *regularizing* conduct. He writes:

Shift the object and change the scale. Define new tactics in order to reach a target that is now more subtle but also more widely spread in the social body. Find new techniques for adjusting punishment to it and for adapting its effects. Lay down new principles for regularizing, refining, universalizing the art of punishing (Foucault 1977, 89)

Foucault’s description of the generalization of the techniques of disciplinary power, the formation that emerged in the late 18th century as a way of ordering ‘massive or transient pluralities’ (143) in Western European societies, seems a long way from microarray gene expression data. Yet the data in many genomic and other settings (transactions, social media, etc.) displays some of the traits – massive, transient, plural – that Foucault identifies as key targets of regulation for the operations of disciplinary power.

The ‘target,’ a term often used in machine learning to describe the type, classification!as ranking group or response being modelled, in genomics is often subtle variations (in phylogeny, in pathogenesis), and these variations are widely dispersed in genomic data and the populations it stems from. Foucault’s account of supervision (*surveiller*) and penalisation as responses to ‘popular illegality’ (130) stresses the capillary network of observations, examining, ranking, test and gradation that adapt to the surging multiplicities by ordering them in tables. While the tables of data (see Table ?? in the microarray gene expression datasets suggest the persistence of the same technique of ordering multiplicities through partitioned observations, the *cells* no longer target contain individuals under observation but focus on the attributes of a multiplicity in movement, the human genome for instance in its many functional states.

‘Shift the object and change the scale,’ Foucault writes, in describing how partitions, segmentations, forms of enclosure, and above all, ranked classifications: ‘discipline is an art of rank, a technique for the transformation of arrangements’ (145). ‘Regularize in a way that automatically drops out features that are not contributing to the class predictions,’ Hastie and co-authors write (Hastie, Tibshirani, and Friedman 2009, 652) in describing how they deal with some of the problems of too many variables in the microarray datasets. The new principles of regularizing in machine learning involve (as we have seen in chapter 4) ‘cost functions’ or techniques of penalization that suppress model complexity, and that train models to eliminate features that yield only low predictive weight. In the many different techniques that *Elements of Statistical Learning* brings to bear on the problem of gene expression – diagonal linear discriminant analysis, nearest shrunken centroids, linear classifiers with quadratic regularization, regularized discriminant analysis, regularized multinomial logistic regression,

machine learning!regularization support vector classifier – essentially the same ordering movement occurs as a model is fit. Regularization re-scales the hyperobject – the patterns of expression of genes associated with different types of tumours – by shrinking or dropping the weights of parameters of each gene in the model and examining the effect on the predictions that result. The coefficients or weights of parameters in the model, the β_p values, are either reduced (L_2 regularization) or eliminated (L_1 regularization) if they contribute little to the predictive accuracy of the machine learner. Learning here takes the form of regularization.

The somewhat daunting *optimization function* for one commonly used machine learning technique called ‘lasso regression’ is frequently used in genomics. It displays features that might help us grasp how machine learners traverse genomic data. Remember that the linear regression model with its diagonal line or plane running through common vector space provides the underlying intuition for many machine learners. We have seen the function in Equation ?? several times already in different variations, including logistic regression used for classification of types or groupings.

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^p X_j \hat{\beta}_j \quad (7.1)$$

In gene expression models, the values of β shown in equation 7.1 map on to the different levels of expression of the many genes indexed by the columns of the microarray dataset. The model tests whether different patterns of gene expression associated with different tumour types. As we have already seen, the number of combinations of genes associated with different tumour types vastly outweighs the number of samples. Regularization is the process of removing or eliminating many of the possible combinations by limiting the values of β that contribute little to the prediction. This is a kind of

'shifting' or 're-scaling' of the object through a training technique. How does this training take place?

The regularized version of the linear regression framework known as 'lasso'

- Least Absolute Shrinkage and Selection Operator – hinges on the lasso penalty shown in equation 7.2¹⁰

$$\hat{\beta}^{lasso} = \operatorname{argmin}_{\beta} \left\{ \frac{1}{2} \sum_i^N (y_i - \beta_0 - \sum_{j=1}^p x_{ij} \beta_j)^2 + \lambda \sum_{j=1}^p |\beta_j| \right\}. \quad (7.2)$$

(68)

Equation 7.2 is notable for the way that it subjects the familiar 'residual sum of squares' way of calculating the coefficients to the 'penalty' carried by the last part of the equation $\sum_i^p |\beta_j|$. As Hastie and co-authors write, 'the lasso does a kind of continuous subset [feature] selection' (69). As always $\operatorname{argmin}_{\beta}$ suggests that the algorithm should find the set of values for β that minimize the overall value of the function. This is a balancing act in this case between reducing the sum of residuals shown in the first half of the equation, and minimizing the sum of the absolute values of β_j in the second part of the function. The overall movements of this feature selection are not difficult to understand, but the point is that the re-shaping of the object proceeds along a diagonal line drawn as the algorithm gradually introduces and scales all of the features in the common vector space \mathbf{X} , only allowing those variables or features to remain in the set that help minimize the difference between the predicted response and the actual response.

10. The original publication of the lasso technique in a paper entitled 'Shrinkage and Selection via the Lasso' (Tibshirani 1996) has been heavily cited in subsequent literature. Google Scholar counts around 13,000 citations. For a paper published in the *Journal of the Royal Statistical Society*, this is surprisingly high, but attests, I would suggest, to the intense interest in renovating linear models for new problems such as image recognition or tumour classification. Somewhat surprisingly, given its heavy usage in other scientists, Andrew Ng's CS229 machine learning course at Stanford University doesn't mention the lasso.

machine learner!Least Absolute Shrinkage and Selection Operator machine

learner!Ordinary Least Sum of Squares

diagrammatic!diagonal

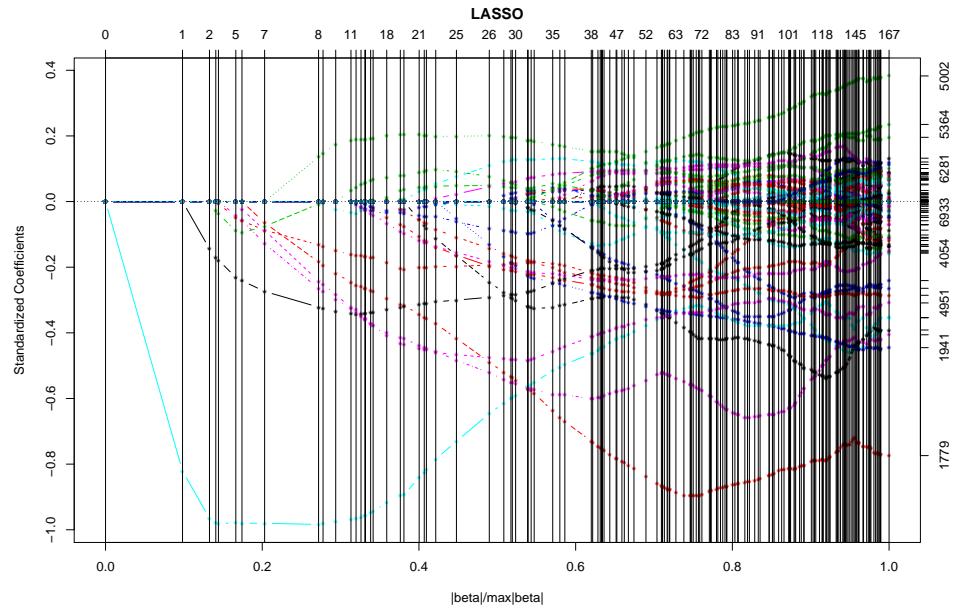


Figure 7.3: Shrinkage path of coefficients for Lasso regression on (Golub,1999) leukemia data

Figure 7.3 makes something of this scaling diagrammatically visible. In this diagram, the various diagonal lines show how values of coefficients grow and sometimes diminish as the `lasso` process runs. Vertical lines show steps as new variables are included in the model with different values of the control parameter λ .

This intensive testing and selection of features results sometimes radically changes the object. In Figure 7.3, these changes become a matter of diagrammatic observation. Comparing eight different methods for analyzing the microarray cancer data from (Ramaswamy et al. 2001), Hastie reports that ‘lasso regression (one versus all)’ selects 1,429 of the 16,063 genes in the dataset. The shifted-rescaled object, a set of 1400 genes, or in the case of the ‘elastic-net penalized multinomial’ model that uses only 384 genes, highlights a drastically reduced subset of the original object.

The proliferation of discoveries

Despite all the penalization and regularization, machine learning does not stabilise genomes as data objects. In many ways, it gives rise to further difficulties, new sources of error and sometimes new transformations of objects.¹¹ If on the one hand, machine learners offer to regularize transient multiplicities (such as epistasis in complex genetic disorders), the observation and supervision of machine learners themselves has become necessary in order to validate and normalize their power in clinical practice, in diagnostic settings or in drug research. Within genomics itself, machine learning figures as a way of correcting for the fact that genomic instruments often produces different results or that genomic knowledge claims shift unpredictably.

For instance, since microarray data itself suffers from many such problems of variation, the US Food and Drug Administration has since 2003 conducted a study of data analysis techniques for microarrays:

The US Food and Drug Administration MicroArray Quality Control (MAQC) project is a community-wide effort to analyze the technical performance and practical use of emerging biomarker technologies (such as DNA microarrays, genome-wide association studies and next generation sequencing) for clinical application and risk/safety assessment (Parry et al. 2010, 292).

11. One important difficulty is the increasingly visible presence of variations in genomes. These variations first become visible after the assembly of whole genome sequences. Genomes of individuals of the same species vary in having slightly different versions of the genes (alleles), many of which differ only by single nucleotide base pairs. Whole genome sequencing made these differences, known as single nucleotide polymorphisms (SNPs), much more apparent. They occur in their tens of millions in the human genome (some 100 million are reported in the NCBI dbSNP database). In coding regions, SNPs may occasion changes in protein structure; in non-coding regions that can affect how gene expression occurs or is regulated. Like genes, SNPs can be detected using microarrays. SNP microarrays are commonly used in genome-wide association studies (GWAS) that explore complex genetic traits and conditions. SNP-based DNA microarrays measure the occurrence of millions of SNPs in a given biological sample.

```
data!variations  
dataset!MACQ-II  
machine learner!_k_  
nearest neighbours  
(KNN)
```

Phase I of the US Federal Drug Administration-led MACQ addressed many issues of data analysis in the context of the clinical applications of gene expression analysis using microarrays. The primary statistical issue there was minimizing the ‘false discovery rate’ (Slikker 2010, S1), a typical biostatistical problem. In MACQ-II however the focus rested on the construction of predictive models for ‘toxicological and clinical endpoints ... and the impact of different methods for analyzing GWAS data’ (2). On both the clinical and GWAS fronts, the 36 participating research teams tried out many predictive classifier models. Different machine learning techniques generate different kinds of models, genomic and biomedical researchers are compelled to engage with the many variations in the data.

Yet, as we have seen, machine learners themselves juggle major sources of errors (for instance, in the bias-variance tradeoff). In the shift from MACQ-I to MACQ-II, the problem of variations in the predictions produced by the machine learning models moved to center-stage. The problem of variation arises not because any of the different modelling strategies used in machine learning gene expression datasets are wrong or erroneous, but because every model moves through the ‘feature space’ (Parry et al. 2010, 292) in a different way (as we saw in chapter 6 in discussions of different treatments of dimensionality). In the MACQ-II consortium, the teams were tasked to build ‘classifiers’ to predict whether a given sample or case belongs to a ‘normal’ or ‘disease’ group. The most popular classifier in the MACQ consortium was the k nearest neighbours model: ‘[a]mong the 19,779 classification models submitted by 36 teams, 9742 were k -nearest neighbor-based (KNN-based) models (that is, 49.3% of the total) (293). But, these models varied greatly in their predictions: ’there have been large variations in prediction performance among KNN models submitted by different teams’ (293). Not only the genome itself varies, but the machine

learners vary.

What accounts for this variation? First of all, the 33 teams did not build single models. As is the norm in machine learning, they built thousands. In their attempt to normalise the variations of their models, one of the research groups in MACQ-II write that ‘for clinical end points and controls from breast cancer, neuroblastoma and multiple myeloma, we systematically generated 463,320 k -nn [k -nearest neighbour] models by varying feature ranking method, number of features, distance metric, number of neighbors, vote weighting and decision threshold’ (292). The striking feature here is the proliferation of models in an effort to tame the variations of predictive models. The number of predictive models constructed here rivals the number of SNPs typically assayed by the microarrays. It seems as if not only the dimensions of the data have vastly increased, but the population of models. All data analysis faces the so-called ‘curse of dimensionality’ (Hastie, Tibshirani, and Friedman 2009, 22), but genomic data is particularly ‘cursed’ by a high model dimensionality.

machine
learning!error!bias-variance
common vector space!di-
mensionality!curse
of
machine learner!_k_
nearest neighbours
Fix, Evelyn

Variations in the object or in the machine learner: k nearest neighbours models

‘The method of k -nearest neighbors makes very mild structural assumptions: its predictions are often accurate but can be unstable’ write Hastie and co-authors (23). The algorithm, first described by Evelyn Fix and Joseph Hodges working at Berkeley in the early 1950s (Fix and Hodges 1951), is extremely simple in mathematical terms.¹² Equation 7.3 shows almost the

12. Fix and Hodges frame their suggestion of the k nearest neighbour model in this way: ‘there seems to be a need for discriminative procedures whose validity does not require the amount of knowledge implied by the normality assumption, the homoscedastic assumption, or any assumption of parametric form. The present paper is, as far as the authors are aware, the first one to attack subproblem (iii): can reasonable discrimination procedures be found which will work even if no parametric form can be assumed?’ (Fix

Choose k , a positive integer which is large but small compared to the sample sizes. Specify a metric in the sample space, for example ordinary Euclidean distance. Pool the two samples and find, of the k values in the pooled samples which are nearest to z , the number M which are X 's. Let $N = k-M$ be the number which are Y 's. Proceed with the likelihood ratio discrimination, using however $\frac{M}{k}$ in place of $f(z)$ and $\frac{N}{k}$ in place of $g(z)$. That is, assign z to F if and only if

$$\frac{M}{k} < c \frac{N}{k}.$$

Figure 7.4: The earliest formulation of the *k nearestneighboursmodel* from Evelyn Fix and Joseph Hodges work [@Fix1951]

entire algorithm:

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i \quad (7.3)$$

'where $N_k(x)$ is the neighbourhood of x defined by the k closest points x_i in the training sample' (Hastie, Tibshirani, and Friedman 2009, 14). The algorithm effectively takes the average values of points in the neighbourhood, and uses that value to predict the result for a given set of features. As Hastie and co-authors put it, the neighbourhood is just those k points near the case under consideration. The assumption here, as in nearly all machine learners traversing the common vector-space, is that proximity implies similarity. This assumption was formally described in the late 1960s in another highly cited paper (Cover and Hart 1967) on 'Nearest Neighbour Pattern Classification.' Neighbouring points in the vector-space are more similar than those at a distance. As equation 7.3 shows, k nearest neighbours seems to have only one parameter, the value k , the number of neighbours that a given model includes in its definition of a 'neighbourhood.'

and Hodges 1951, 7). Subproblem (iii) in this quote refers to the challenge of deciding which of two populations an observed case belongs to if we know nothing about the parameters describing the two populations.

VARIATIONS IN THE OBJECT OR IN THE MACHINE LEARNER: K NEAREST NEIGHBOURS MODELS

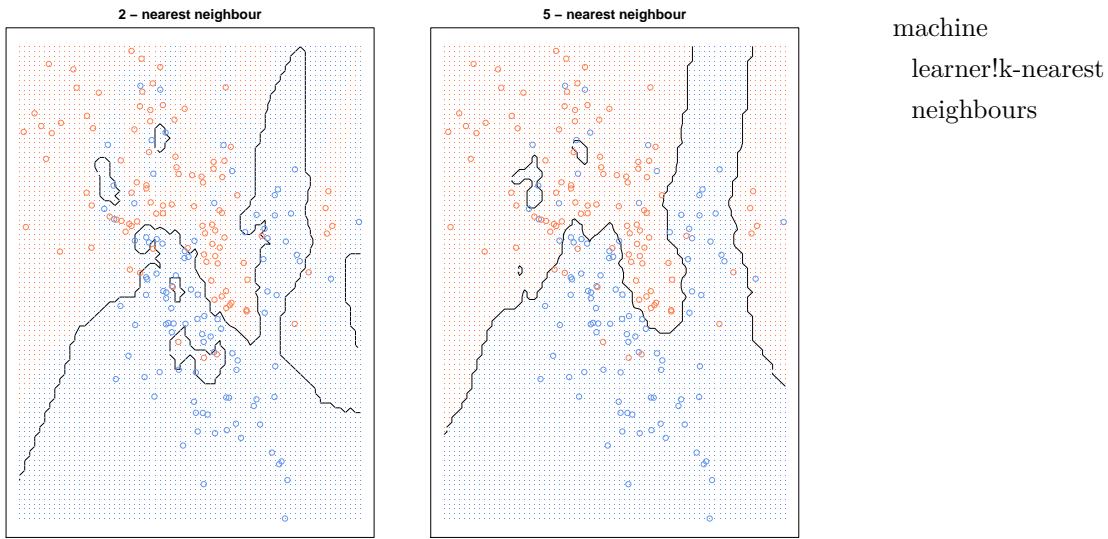


Figure 7.5: KNN models for simulated data in two classes. The decision boundary that separates the two classes is non-linear for both versions of the KNN model. For $k = 5$, the decision boundary is more non-linear than for $k = b$

In contrast to the linear forms of the models (formulated in equations 7.2 or 8.2), equation 7.3 seems to require little training, supervision or regularization to work as a classifier. While nearly all of the models I have discussed in this and earlier chapters still work with the smooth form of the line or curve as their basic way of separating or connecting, k nearest neighbours models easily generate highly non-linear boundaries wending their way through the data. Because they are not guided by parameters (apart from the value of k), these boundaries can be unstable.

Most machine learning techniques embody some way of managing the dimensionality of the feature space, either by effectively reducing that dimensionality (as we saw in the lasso technique, which corrals the numbers of predictive features in the model), or concentrating on localised regions of the feature space, as does k -nn. For instance, the k -nn models — ‘ k -nearest neighbour models’ — analyzed by Parry as part of MACQ-II are widely used because they are the simplest way of dealing with interactions in an ‘exceedingly large feature space’ (Parry et al. 2010, 292). In k -nn, a training

set of observations whose outcomes are known (e.g. clinical endpoints in cancer might be benign or malignant) are used to help classify (and hence predict) the test data.

As we can see from Figure 7.5 above, in which data belongs to two classes (normal vs. not-normal), the decision boundaries produced by the algorithm can be unstable. The example shows two models, one for $k = 5$ and the other for $k = b$. Each model examines the relations between 2, 5 points in deciding whether a particular case belongs to one class or another. While $k\text{-}nn$ finds local clusters and classifies on the basis of an irregular decision boundary, this classificatory power comes at the cost of instability. The more dimensions or features in the dataset, the larger the local neighbourhood needed to capture a fraction of the volume of the data, and the more likely that most sample points will lie close to the boundary of the sample space, where they will be affected by the neighbouring space. The result is that ‘in high dimensions all feasible training samples sparsely populate the input space’ (Hastie, Tibshirani, and Friedman 2009, 23). Because $k\text{-}nn$ allows for non-linear interactions between features, for instance, small differences in the number of points in particular neighbourhoods can drastically affect the boundaries (as we see in comparing the right and left hand plots). These kinds of topological instability account for the propensity of machine learning treatments of feature-rich genomic data to produce accurate but unstable predictions. We can also see why the MACQ-II team reported in (Parry et al. 2010) might have ended up producing 463,000 k nearest neighbour models in an effort to normalise and regulate predictive predictions. The price of accurate predictivity in genomics is variation in prediction.

Whole genome functions

Chip cores, SNPs, and models; infrastructural scaling, biological variation and the training of machine learners entwined in a referential entanglement. If genomes are scientific hyperobjects (with all the epistemic, speculative, financial and biopolitical resonances), what part does machine learning play in the transformation of science?

Science is concentrated in an area of knowledge it does not absorb and in a formation which is in itself the object of knowledge and not of science. Knowledge (*savoir*) in general is not science or a even a particular corpus of knowledge (Deleuze 1988, 19)

I have been suggesting that the genomic data – beginning with DNA sequences, then levels of gene expression, followed by genome wide association studies of small mutations, and finally by whole genome analyses – has been a constant $p \gg N$ antigen in machine learning. It is no accident that the techniques of regularization – the lasso – of linear models discussed in this chapter came to light, and were first demonstrated on the genomic data produced in the mid-1990s. Throughout the ongoing development and enrichment of DNA and protein sequencing techniques, replete with a vast and quite dynamic bioinformatic infrastructure, machine learning and genomics have been in a reciprocal embrace. Scientists and statisticians traffic between genomics and machine learning at almost every level, ranging from the sequence assembly to testing and analysis of DNA data in clinical settings. For genomics, elementary practices of aligning and assembling sequences into whole genomes were re-configured probabilistically through machine learning models. Almost every subsequent development in genomics and related fields such as proteomics follows this pattern: an entity whose constitution is thoroughly dependent on prediction or algorithmic

machine learning!positivity of classification displays variations and grouping (such as gene expression, the linkage disequilibrium of SNPs, the seeming abundance of junk DNA that is actually functioning, etc.) that attract further efforts to differentiate and classify ever more subtly distributed differences. It is hard to imagine contemporary biomedicine, biotechnology, pharmaceutical and environmental science research without this pattern. Elementary practices in contemporary genomics such as sequence alignment were explicitly formulated as generative models to be constructed using algorithms such as expectation maximization. Cancer, in all its uncontrolled, diffuse and dangerously tumid concentrations, is the target of the most concerted ranking and typing treatments. As we see in the vignettes from *Elements of Statistical Learning*, the demonstration of Google Compute cloud computing, or for that matter in the myriad publications in both machine learning and life science journals that make use of support vector machines, neural networks, linear discriminant analysis or random forests, the positivity of machine learning establishes a new set of conditions for the exercise of scientific research, and configures new kinds of statements, new fields of objects (genomes in particular are difficult to conceive without their probabilistic modelling) and, as will be discussed in the next chapter, subject positions (bioinformaticians, computational biologists, data scientists and others).

The machine-learned models flow back through genomic science, reorganising and refocusing research, experiments, databases and funding in subtle but important ways. Between pre-genomic and post-genomic science, the status of significant differences in genomes shifted. Pre-HGP biology understood the significant differences between individual organisms largely in terms of gene alleles responsible for variations in phenotypes. Biological differences, and disease in particular, stemmed from different forms of genes. Understanding disease meant finding the disease genes. Even

prominent proponents of genomics, such as Leroy Hood, writing of ‘Biology data!sequence and Medicine in the Twenty-First Century’ in 1991, envisaged genomics as a way of simplifying ‘the task of finding disease genes’ (Hood and Kevles 1992, 138). Across the life sciences, genes were the object of much way of annotation, labelling and description. Two decades after the inception of whole genome sequencing, genomes present a different image of variation. According to Nikolas Rose, writing more recently, ‘there is no normal human genome; variation is the norm’ (Rose 2009, 75). ‘In this new configuration’, he writes, ‘what is required is not a binary judgment of normality and pathology, but a constant modulation of the relations between biology and forms of life, in the light of genomic knowledge.’ The emphasis in Rose’s formulation falls on ‘constant modulation’ of the relations between biology and forms of life. If post-genomic science departs from the understanding that there is no single genome but many genomes, then according to Rose, variation itself becomes of primary interest. Pursuit of variation remakes the genome into ‘a form whose only object is the inseparability of distinct variations’ (Deleuze and Guattari 1994, 21).

Machine learning widely affects the forms of accumulation, the enunciative modalities, and the ordered multiplicities that constitute scientific knowledges. In the biosciences of the last two decades, it seeks to disaggregate, compartmentalise and rank those aspects of genomes — their confused variations, their manifold spatial and temporal relationality in biological processes — that seem most distant and difficult to derive from the putatively linear, monolithic and hence tractable order of DNA sequence data. As we have seen in the various data excerpts above, that order is largely linear. DNA can be laid down in tracks or grids, aligned and annotated in uniquely addressable database records, but the problem of how this extensive vector maps onto the subtle, pervasive and transient forms of temporal and

common vector space!dimensionality!curse of spatial re-shaping in life-forms remains. None of the examples of genomic data in *Elements of Statistical Learning* use whole genome sequences. In

the feature-rich spaces countenanced by machine learning, we see attempts to embed manifolds in local regions, local linearities. Sometimes these local regions are regions of annotated DNA, as in the ENCODE examples, or non-linear interactions between sets of genes, as in the GWAS analysis of epistasis. At other times, these local regions are forms of life in a more general sense — clinical outcomes or diagnostic tests — as in MACQ-II.

I have emphasised on several occasions the common thread of what could be termed the ‘genomic curse of dimensionality.’ The genomic sciences are based around the extraction, sorting and ordering of DNA sequences. Machine learning practices, I have suggested, begin to construct different dimensional spaces amidst the sequences, lines and tracks of genomes. What perhaps most vexes and animates post-genomic science is the desire to separate out from the DNA sequences variations that matter to both life-forms and forms of life. This vexation generates strenuous infrastructural, technical and conceptual attempts to reorganise and re-conceptualise what it is to know a genome. The machine learning techniques I have discussed have begun to transform bioinformatics and biostatistics. To understand machine learning as either as intensification of statistics, as hybridization of bioinformatics and statistics, or as the product of a reciprocal interactive convergence of biology and computing misses some of the important features of the reorganisation of genomics associated with these practices. It misses what machine learning brings from elsewhere to genomics, and it overlooks how the genome itself serves as a showcase, as we saw in the Google I/O demonstration, of certain much wider transformations in the practices of anticipation and prediction.

What is at stake in approaching machine learning in a scientific setting like genomics? An analytical and an ethico-epistemic stake appear here. Foucault writes that ‘we should distinguish carefully between *scientific domains* and *archaeological territories*’ (Foucault 1972, 183). If knowledge has the status of knowledge by virtue of the practices that connects objects, field, subjects, statements, and institutions, then sciences are always localized within a field of knowledge that may exceed, and that may mutate in ways that alter resident sciences. Machine learning is just such a formation, I would suggest. Could we pose or address any normative questions by becoming aware of and articulating machine learning in practice with greater clarity? Genomic science, in its cross-validation with machine learning, displays some of the tendencies to reduce divergences and to corral differences typical of knowledge economies more generally. The philosopher of science, Isabelle Stengers writes:

with the knowledge economy, we may have scientists at work everywhere, producing facts with the speed that new sophisticated instruments make possible, but that the way those facts are interpreted will now mostly follow the landscape of settled interests. ... We will more and more deal with instrumental knowledge. (Stengers 2011:377)

As we see in the 600,000 cores of Google Compute applied to exploration of associations in cancer genomics using random forests, or the lasso applied to microarray SNP data, machine learning rapidly produces facts. Stengers suggests that the risk here is that divergence and unexpected forms of experimental result are somewhat diminished as a result. Machine-learning in genomics might produce a ‘self-organising map’ that poses questions following the ‘landscape of settled interests’ or *status quo*.

The very justification for using such techniques is the inordinate difficulty of exploring the functioning of genomes otherwise. Genomes and genomics are touchstones for wider transformations in many sectors of science, industry, commerce, media and government susceptible to the imperatives of the knowledge economy. In contrast to some of these domains, where much that happens is obscured from view, the great virtue of genomic science is the relative openness of its workings and its dogged insistence on DNA as the generating set. The fact that data practices are relatively generic and accessible means that critical research into transformations associated with data and knowledge economies can accompany nearly every aspect of genomic practice.

statement!position of
subject

Chapter 8

Propagating subject positions

If a proposition, a sentence, a group of signs can be called ‘statement’ , it is not therefore because, one day, someone happened to speak them or put them into some concrete form of writing; it is because the position of the subject can be assigned (95)

‘generalization error is what we care about’ (*Lecture 9 / Machine Learning (Stanford) 2008*)

Predict if an online bid is made by a machine or a human,
‘Facebook Recruiting IV: Human or Robot?’ (Kaggle 2015d)

In early 2002, while carrying out an ethnographic study of ‘extreme programming,’ a software development methodology popular at that time (Mackenzie and Monk 2004), I spent several months visiting a company in Manchester developing software for call centres. The purpose of the software was to manage ‘knowledge’ in call centres such that any query from a caller would be readily answered by call centre staff who would query a knowledge

machine learner!neural
net
kittydar

management system. This system was marketed on the promise of machine learning. It contained an artificial neural network that learned to match queries and responses over time. A taciturn neural network expert, Vlad, sat in a different part of the room from the developers working on the databases and the web interfaces of the knowledge management system. Vlad's work on the neural network was at the core of the knowledge management system yet outside the orbit of the software development team and its agile software development processes. They generally regarded Vlad and the neural net as an esoteric, temperamental yet powerful component, a hidden node we might say, of the knowledge management system.

As we have already seen with `kittydar`, today the situation of machine learners has changed. They are no longer exotic or specialized, but both banal and occasionally spectacular. Hilary Mason, who was Chief Scientist at bitly.com (an online service that shortens URLs), outlined a machine learning subject position at a London conference in 2012 called 'Bacon: Things Developers Love':

You have all of these things that are different – engineering, infrastructure, mathematics and computer science, curiosity and an understanding of human behaviour – that is something that usually falls under the social sciences. At the intersection of all these things are wonderful people. But we're starting to develop something new, and that is - not that all of these things have not been done for a very long time - but we are only just now building systems where people, individual people, have all of these tools in one package, in one mind. And there are lots of reasons this is happening now. But its a pretty exciting time to be in any of these things. (Hilary Mason, Chief Scientist, bitly.com) (**Bacon_2012**)

These ‘things that are different,’ if they amount to a *statement* in Michel Foucault’s sense of that term, together assign subject positions . In what ways? In front of an audience of several hundred software developers, Mason describes shifts in the work of programming associated with the growth of large amounts of data associated with ‘human behaviour.’ At the centre of this shift stand ‘wonderful people’ who combine practices and knowledges of communication infrastructure, technology, statistics, and ‘human behaviour’ through curiosity and technical skills. Mason was, in effect, telling her audience of software developers who they should become and at the same time pointing to the some of the expansive changes occurring around them. The title of her talk was ‘machine learning for hackers’, and her audience were those hackers or programmers whose coding and programming attention may have been previously trained on web interfaces or database queries, but was now drawn towards machine learning. A change in programming practice and a shift towards machine learning was, she implied, the key to programmers becoming the wonderful people, agents of their own time, capable of doing what is only now just possible because it is all together in ‘one package, one mind.’ Note that concatenation of ‘one package, one mind’ does not definitively allocate agency to people or things. Here, I would tentatively suggest, Mason adumbrates the outline of an agent of anticipation, someone or something at the intersection of network infrastructure, mathematics and human behaviour.¹ Mason, one of *Fortune* magazines ‘Top 40 under 40’ business leaders to watch (CNN 2011) and also featured in *Glamour*, a teenage fashion magazine (*Hilary Mason*

1. In earlier work on machine learning (Mackenzie 2013c), I presented programmers as agents of anticipation, suggesting that the turn to machine learning amongst programmers could be useful in understanding how predictivity was being done amidst broader shift to the regime of anticipation described by Vincenne Adams, Michelle Murphy and Adele Clarke (Adams, Murphy, and Clarke 2009). Subsequently developments in machine learning, even just in the last three years, confirm that view, but in this chapter and in this book more generally, I focus less on transformations in programming practice and software development, and more on the asymmetries of different machine learner subjects in relation to infrastructures and knowledge.

Mason, Hilary

Suchman, Lucy

Ng, Andrew

- *Machine Learning for Hackers* 2012), might personify such a wonderful person. She is not a lone example.² Equally so, she might *a-personify* the intersection.

'It is the privileged machine in this context that creates its marginalized human others' writes Lucy Suchman in her account of the encounters that 'effect "persons" and "machines" as distinct entities' (L. Suchman 2006, 269) . While Mason and other relatively well-known human machine learners are not exactly marginalized (just the opposite, they achieve minor celebrity status in some cases), Suchman recommends 'recognition of the particularities of bodies and artifacts, of the cultural-historical practices through which human-machine differences are (re-)iteratively drawn, and of the possibilities for and politics of redistribution across the human machine boundary' (285). The intersections that machine learners currently occupy are heavily re-distributional. In almost every instance, machine learners claim to do something that humans alone, no matter how expert, could not. (We need only think of the demonstration of the radio-controlled helicopter flying upside down that Andrew Ng shows in his introduction of machine learning lecture (*Lecture 1 / Machine Learning (Stanford)* 2008) .) Could the 'wonderful people' that Mason describes also be seen as marginalized human others? Does the re-distribution of engineering, mathematics, curiosity, infrastructure and 'something that usually falls under the social sciences' (but perhaps no longer does so?) both energise subjects ('its a pretty exciting time to be in any of these things') and assign them a marginal albeit still pivotal position?

Machine learner subject positions are the topic of this chapter. I draw on artificial neural networks, or neural nets, in their various forms ranging from

2. Other figures we might follow include Claudia Perlich, Andrew Ng, Geoffrey Hinton, Corinna Cortez, Daphne Koller, Christopher Bishop, Yann LeCun, or Jeff Hammerbacher. Although some women's names appear here, in any such list, men's names are much more likely to appear. This is no accident.

the perceptron, through the multilayer perceptron (MLP), the convolutional neural nets (CNN), recurrent neural nets (RNN) and deep belief networks or deep neural networks of many recent deep learning projects (particularly in machine learning competitions, as discussed below (Dahl 2013)). In exploring the re-drawing of subject-machine positions. Neural nets propagate between infrastructures, engineering and human behaviour (as Mason puts it), re-drawing human-machine differences. A straightforward example of this re-drawing appears in neural net publications. Geoffrey Hinton, Simon Osindero and Yee-Why Teh writing in *Neural Computation* in 2006 described a ‘fast learning algorithm for deep belief nets’ (Hinton, Osindero, and Teh 2006). Their description, whilst mostly couched in terms of conditional distributions, parameters, and generative models also contains a section entitled ‘Looking into the Mind of a Neural Network’ (1545-1546). In that section, they describe how they used their deep belief network to *generate* rather than classify images.³ In the process they were able to see what the ‘associative memory has in mind’ (1545). The term ‘mind’ it turns out ‘is not intended to be metaphorical’ (1546) because the neural net in question has a distributed memory of the digits it has seen. Put slightly more formally, ‘the network has a full generative model, so it is easy to look into its mind - we simply generate an image from its high-level representations’ (1529). The conjunction or intersection of ‘mind’ and generative model draws a new diagram of artificial intelligence (which has typically relied on rule-based or symbolic reasoning), but the appearance of ‘mind’ in the form of a generative model (see chapter 5) covers over other layers of positioning and subjectification. These layers might be more interesting to explore archaeologically (for instance, in the work done on specific artifacts such as images) than in terms of the general existential threat of artificial

neural net|seemachine
learner!neural net
diagram!machine learner
as
Hinton, Geoffrey
model!generative

3. In the case of this paper, and many others related to neural nets, the images are of hand-written digits. These digits have an almost constitutive role, as I discuss in this chapter.

back-propagation!intelligence Neural nets re-iterate human machine boundaries through a function!enunciative!of combination of feeding-forward of potentials and propagating backwards neural net of differences specifically concerned with images. Similarly, the practice of machine learning assigns subject positions in a backward and forward movement that propagates potentialising optimism even as it undercuts the very differences that give rise to that optimism.

Propagating neural nets in machine learning

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

Almost every machine learning class, textbook, demonstration, and in recent years, in machine learning competitions at some point turns to neural nets. They display, however, some instability in the research literature. Table 8.1 shows the most frequent keywords for technical publications across the three main disciplinary domains inhabited by machine learners. While neural nets rank very high in computer science and engineering disciplines (appearing just after support vector machines), they do not appear in the statistics literature until 33 in the rankings. The prominence of neural nets on the engineering side of machine learning suggests they perform specific enunciative functions. Neural nets are often described from a deeply split perspective that in some points turns towards human subjects, or at least, the brains of human subjects, and in other ways towards the ongoing expansion of computational platforms. In some ways, they renew long-standing cybernetic hopes of brains and cognition as models of computational intelligence and agency. Although they stem from a biological inspiration (dating at least back to the work on McCulloch and Pitts in the 1940s (Halpern 2015; Wilson 2010)), they gain traction first in the 1980s and then again from mid-2000s onwards, as ways of dealing with

changing computational infrastructures, and the difficulties of capitalising on infrastructure that is powerful yet difficult to manage. Neural nets almost constantly oscillate between brain and information infrastructure. In the course of fifty years, their triple re-invention – from perceptron via neural net to deep belief net – re-distributes subject positions amidst infrastructural re-configurations and vectorisation. For instance, David Ackley, Geoffrey Hinton (an important figure in the inception of neural nets during the 1980s and in the revival of neural net in the form of deep learning in the last decade), and Terrence Sejnowski wrote in the early 1980s:

Evidence about the architecture of the brain and the potential of the new VLSI technology have led to a resurgence of interest in “connectionist” systems ... that store their long-term knowledge as the strengths of the connections between simple neuron-like processing elements. These networks are clearly suited to tasks like vision that can be performed efficiently in parallel networks which have physical connections in just the places where processes need to communicate. ... The more difficult problem is to discover parallel organizations that do not require so much problem-dependent information to be built into the architecture of the network. Ideally, such a system would adapt a given structure of processors and communication paths to whatever problem it was faced with (Ackley, Hinton, and Sejnowski 1985, 147-148).

Alignments and diagrammatic overlaps between brain and ‘new VSLI [Very Large Scale Integrated] technology’ – semiconductor chips – architectures sought to map the plasticity of neuronal networks onto the parallel distributed processing enabled by very densely packed semiconductor circuits. The problem here was how to organize these connections without having

infrastructure!reconfigura-
ration
of
vectorisation!of
infrastructure

Hinton, Geoffrey
diagram!overlay

machine to hardwire domain specificity into ‘the architecture of the network.’ How learner!perceptron could the architectures adapt to the problem in hand?

algorithm!back-propagation

programmability!problem of!neural net as solution to We saw in chapter 2 that the psychologist Frank Rosenblatt’s perceptron (Rosenblatt 1958) first implemented McCulloch and Pitts’ cybernetic vision of neurones as models of computation (Edwards 1996). While the perceptron did not weather the criticism of artificial intelligence experts such as Marvin Minsky (Minsky famously showed that a perceptron cannot learn the logical exclusive OR or XOR function; (Minsky and Papert 1969)), cognitive psychologists such as David Rumelhart, Geoffrey Hinton and Ronald Williams persisted with perceptrons, seeking to generalize their operations by connecting them together in networks (also known as multi-layer perceptrons). In the mid-1980s, they developed the back-propagation algorithm (Rumelhart, Hinton, and Williams 1985; Hinton 1989), a way of adjusting the connections between nodes (neurones) in the network in response to features in the data (see Figure 8.1). The back-propagation algorithm directly addressed the problem of modifying parallel network organizations without reliance on problem-dependent architectures, and in without having to program them in. Effectively, an architecture of generalization was implemented. While cognition, and the idea that machines would be cognitive (rather than say, mechanical, calculative, or even algorithmic) constantly organised research work in artificial intelligence for several decades, the development of the back-propagation algorithm as a way for a set of connected computational nodes to learn also had strong infrastructural resonances. These resonances became much more visible from around 2006 when ‘deep belief nets’ appeared as a way of training many-layered neural nets (Hinton_2006b). These resonances continue to echo today and indeed attract much attention.⁴ Like the advent of VSLI

4. Although mainstream media accounts of machine learning are not the focus of my interest here, it is hard to ignore the extraordinary level of interest that deep learning

Unclassified SECURITY CLASSIFICATION OF THIS PAGE		
REPORT DOCUMENTATION PAGE		
1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE		
4. PERFORMING ORGANIZATION REPORT NUMBER(S) ICS 8506	5. MONITORING ORGANIZATION REPORT NUMBER(S)	
6a. NAME OF PERFORMING ORGANIZATION Institute for Cognitive Science	6b. OFFICE SYMBOL (if applicable)	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) C-015 University of California, San Diego La Jolla, CA 92093	7b. ADDRESS (City, State, and ZIP Code)	
8a. NAME OF FUNDING/Sponsoring Organization Personnel & Training Research	8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-85-K-0450
8c. ADDRESS (City, State, and ZIP Code) Code 1142 PT Office of Naval Research 800 N. Quincy St., Arlington, VA 22217-5000	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. PROJECT NO. TASK NO. WORK UNIT ACCESSION NO. NR 667-548	
11. TITLE (Include Security Classification) Learning Internal Representations by Error Propagation		
12. PERSONAL AUTHORS David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams		
13a. TYPE OF REPORT Technical	13b. TIME COVERED FROM Mar 85 TO Sept 85	14. DATE OF REPORT (Year, Month, Day) September 1985
15. PAGE COUNT 34		
16. SUPPLEMENTARY NOTATION To be published in J. L. McClelland, D. E. Rumelhart, & the PDP Research Group, <i>Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Vol 1. Foundations</i> . Cambridge, MA: Bradford Books/MIT Press.		
17. COSATI CODES FIELD GROUP SUB-GROUP		
18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) learning; networks; perceptrons; adaptive systems; learning machines; back propagation		

Figure 8.1: An early publication of the back-propagation algorithm: Rumelhart, Hinton and William's 1985 paper [@Rumelhart_1985]

in the early 1980s, the vast concentrations of processing units in contemporary data centres (hundreds of thousands of cores as we saw in the case of Google Compute in the previous chapter ??) and even in the graphics cards developed for high-end gaming and video rendering (GPUs for PC gaming now typically have a thousand and sometimes several thousand cores) pose the problem of organizing infrastructure so that processes can communicate with each other. Machine learners may well become more important as loose or mutable infrastructural arrangements than as epistemic instruments.

projects and techniques have attracted in the last few years. Articles have appeared in all the usual places – *The New York Times* (Markoff 2012), *Wired* (Garling 2015), or *The Guardian* (C. Arthur 2015). In many of these accounts, machine learning and neural nets in particular appear both in the guise of the existential threat of artificial intelligence and as a mundane device (for instance, speech recognition on a mobile phone). The spectacular character of deep learning could be analysed in terms like that of the genomes discussed in chapter 7. In both cases, the advent and transformation of these machine learners is closely linked to networked platforms (such as Google, Facebook, Yahoo and Microsoft) and their efforts to encompass within their services as many elements of experience, exchange, communication and power as possible. Deep learning machine learners currently focus mostly on images (photographs and video) and sounds (speech and music), and usually attempt to locate and label objects, words or genres.

common vector
space!epistopics

The back-propagation or ‘backprop’ algorithm will return below, but for the moment, this oscillation between cognition and infrastructures, between people and machines, itself suggests another way of thinking about how ‘long-term knowledge’ takes shape today. At the same time as infrastructural reorganization takes place around learning, and around the production of statements by machine learners, both human and non-human machine learners are assigned new positions. These positions are dispersed, hierarchical and distributed. The subject position in machine learning is a highly relational one rather than a single concentrated form of expertise (as we might find in a clinical oncologist, biostatistician or geologist). Because machine learners construct vectorized epistopics, partial observers traversing vector spaces, inverse probabilisations, pattern dispersions and regularizing hyperobjects, what counts as agency, skill, action, experience and learning shifts constantly. It is intimately bound and connected to transforms in infrastructure, variations in referentiality (such as we have seen in the construction of the common vector space), and competing forms of authority (as we have seen in the accumulations of different techniques). As Suchman suggests, examining privileged machines is a way to pay attention to variously marginalized human others.

The ranking of keywords in table ?? suggests that some machine learners attribute a more privileged and constitutive function to neural nets. Neural nets synchronically spread into many difference disciplines: cognitive science, computer science, linguistics, adaptive control engineering, psychology, finance, operations research, etc., and particularly statistics and computer science during the 1980-1990s. This dendritic growth did not just popularise machine learning. It brought engineering and statistics together more strongly. Ethem Alpaydin, a computer scientist, writes:

Perhaps the most important contribution of research on neural

networks is this synergy that bridged various disciplines, especially statistics and engineering. It is thanks to this that the field of machine learning is now well established (Alpaydin 2010, 274).

dataset!MNIST
machine
learning!competitions

The forms of this field-making bridging are various.⁵ The primary space of coexistence of different disciplines has perhaps been the machine learning competitions of the 1990s that pitted neural nets against other machine learners in classifying handwritten numerals such as zipcodes on envelopes, and in particular one set of handwritten digits that remain in constant use, the MNIST (Modified National Institute of Standards) dataset (LeCun and Cortes 2012). The many competitions focused on the MNIST dataset are, I suggest, a form of demonstration and testing of machines and people that propagate machine-human differences in machine learning.

It is no accident that Hastie and co-authors in *Elements of Statistical Learning* devote a lengthy section to the analysis of these handwritten digit recognition competitions that began in the early 1990s and continue today. Like Alpaydin, they affirm the coordinating effect of these competitions on the development of machine learning:

This problem captured the attention of the machine learning and neural network community for many years, and has remained a benchmark problem in the field (Hastie, Tibshirani, and Friedman 2009, 404).

The handwritten digits used in these competitions, particularly the Neural Information Processing System workshops and KDD Cup (Knowledge Discovery and Data Mining) (KDD 2013), all come from the MNIST dataset

5. We saw some use of neural nets in genomics in the previous chapter (7). The initial publication of the SRBCT microarray dataset in (Khan et al. 2001) relied on neural nets.

overfitting
 programmability!problem
 of

and during the 1990s, much effort focused on crafting neural nets to recognise these 60,000 or so handwritten digits. As Hastie and co-authors observe, ‘at this point the digit recognition datasets became test beds for every new learning procedure, and researchers worked hard to drive down the error rates’ (Hastie, Tibshirani, and Friedman 2009, 408-409).

Despite this binding function, neural networks have had a somewhat problematic position in to machine learning. Even in relation to the paradigmatic handwritten digit recognition problem, neural nets struggled to gain purchase. On the one hand, their analogies and figurations as sophisticated neuronal-style models suggested cognitive capacities surpassing the more geometrical, algebraic and statistically grounded machine learners such as linear discriminant analysis, logistic regression, or decision trees. On the other hand, the density and complexity of their architecture made them difficult to train. Neural nets could easily overfit the data. As *Elements of Statistical Learning* puts it, it required ‘pioneering efforts to handcraft the neural network to overcome some these deficiencies..., which ultimately led to the state of the art in neural network performance’ (404). It is rare to find the word ‘handcraft’ in machine learning literature. The operational premise of most machine learners is that machine learning works without handcrafting, or that is automates what had previously been programmed by hand. Somewhat loopily, the competition to automate recognition of handwritten digits, the traces that epitomise movements of hands, entailed much handcrafting and recognition of variations in performances of the machine.

The shifting fortunes of neural nets are frequently discussed in contrasting terms by machine learners themselves.⁶ In recent years, they often point to

6. Neural nets also receive uneven attention in the machine learning literature. In Andrew Ng’s Stanford CS229 lectures from 2007, they receive somewhat short shrift: around 30 minutes of discussion in Lecture 6, in between Naive Bayes classifiers and several weeks of lectures on support vector machines ([Lecture 6 / Machine Learning](#)

some kind of transformation:

Neural networks went out of fashion for a while in the 90s - 2005 because they are hard to train and other techniques like SVMs beat them on some problems. Now people have figured out better methods for training deep neural networks, requiring far fewer problem-specific tweaks. You can use the same pretraining whether you want a neural network to identify whose handwriting it is or if you want to decipher the handwriting, and the same pretraining methods work on very different problems. Neural networks are back in fashion and have been outperforming other methods, and not just in contests ([Zare 2012](#)).

The somewhat vacillating presence of neural nets in the machine learning literature itself finds parallels in the fortunes of individual machine learners.

([Stanford](#) 2008). As he introduces a video of an autonomous vehicle steered by a neural net after a 20 minute training session with a human driver, Ng comments that ‘neural nets were the best for many years.’ The lectures quickly moves on to the successor, support vector machines. In *Elements of Statistical Learning*, a whole chapter appears on the topic, but prefaced by a discussion of the antecedent statistical method of ‘projection pursuit regression.’ The inception of ‘projection pursuit’ is dated to 1974, and thus precedes the 1980s work on neural nets that was to receive so much attention. In *An Introduction to Statistical Learning with Applications in R*, a book whose authors include Hastie and Tibshirani, neural nets are not discussed and indeed not mentioned (James et al. 2013). Textbooks written by computer scientists such as Ethem Alpaydin’s *Introduction to Machine Learning* do usually include at least a chapter on them, sometimes under different titles such as ‘multi-layer perceptrons’ ([Alpaydin 2010](#)). Willi Richert and Luis Pedro Coelho’s *Building Machine Learning Systems with Python* likewise does not mention them ([Richert and Coelho 2013](#)). Cathy O’Neil and Rachel Schutt’s *Doing Data Science* mentions them but does not discuss them ([Schutt and O’Neil 2013](#)), whereas both Brett Lantz’s *Machine Learning with R* ([Lantz 2013](#)) and Matthew Kirk’s *Thoughtful Machine Learning* ([Kirk 2014](#)) devote chapters to them. In the broader cannon of machine learning texts, the computer scientist Christopher Bishop’s heavily cited books on pattern recognition dwell extensively on neural nets (Bishop et al. 1995; Bishop 2006). Amongst statisticians, Brian Ripley’s *Pattern Recognition and Neural Networks* ([Ripley 1996](#)), also highly cited, placed a great deal of emphasis on them. But these specific documents against a pointillistic background of hundreds of thousands of scientific publications mentioning or making use of neural nets since the late 1980s in the usual litany of fields – atmospheric sciences, biosensors, botany, power systems, water resource management, internal medicine, etc. This swollen publication tide attests to some kind of formation or configuration of knowledge invested in these particular techniques, perhaps more so than other I have discussed so far (logistic regression, support vector machine, decision trees, random forests, linear discriminant analysis, etc.).

Le Cun, Yann
 Hinton, Geoffrey
 Cortes, Corinna
 deep
 learning|seemachine
 learner!deep learning
 Ng, Andrew
 kittydar

Yann Le Cun's work on optical character recognition during 1980-1990s is said to have discovered the back-propagation algorithm at the same time as Rumelhart, Hinton and Williams (Rumelhart, Hinton, and Williams 1986). His implementations in **LeNet** won many research machine learning competitions during the 1990s. In 2007, Andrew Ng could casually observe that neural nets *were* the best, but in 2014, Le Cun find himself working on machine learning at Facebook (Gomes 2014). Similarly, the cognitive psychologist Geoffrey Hinton's involvement in the early 1980s work on connectionist learning procedures in neural nets and subsequently on 'deep learning nets' (Hinton and Salakhutdinov 2006) delivers him to Google in 2013. Trajectories between academic research and industry are not unusual for machine learners. Many of the techniques in machine learning have been incorporated into companies later acquired by other larger companies. Even if there is no spin-off company to be acquired, machine learners themselves have been assigned key positions in many industry settings. Corinna Cortes, co-inventor with Vladimir Vapnik of the support vector machine, heads research at Google New York . In 2011, Ng led a neural net-based project at Google that had, among other things, detected cats in millions of hours of Youtube videos.⁷ Ng himself in 2014 began work as chief scientist for the Chinese search engine, Baidu leading a team of AI researchers specializing in 'deep learning,' the contemporary incarnation of neural nets (Hof 2014) . In recent years, (2012-2015), work on neural nets has again intensified, most prominently in association with social media platforms, but also in the increasingly common speech and face recognition systems found in everyday services and devices. Many of these neural nets are like **kittydar** , but implemented on a much larger and more distributed

7. Unlike the cats detected by **kittydar** , the software discussed in the introduction to this book, the Google experiment did not use supervised learning. The deep learning approach was unsupervised (Markoff 2012). That is, the neural nets were not trained using labelled images of cats.

scale (for instance, in classifying videos on Youtube). In contemporary gradient descent machine learning competitions, as we will see, neural nets again surface as intersectional machines, re-distributing differences between humans and machines.

A privileged machine and its diagrammatic forms

What accounts for the somewhat uneven fortunes of the neural net amongst machine learners? The unevenness of their performance, from limited curiosity in the late 1960s to best performer in the machine learning image classification competitions of the 1990s, from second best competitor in late 1990s to the spectacular promise of deep belief networks in 2012, suggests that some powerful dynamics or becomings are in play around them. These dynamics are not easily understood in terms of celebrity machine learners (human and non-human) suddenly rising to prominent or privileged positions in the research departments of social media platforms.⁸ Nor does it make sense to attribute the rising fortunes of the neural net to the algorithms themselves, as if some decisive advance occurred in algorithms. The algorithms used in neural net have not, as we will see, been radically transformed in their core operations since the 1980s, and even then, the algorithms themselves (principally gradient descent) were not new. There have been important changes in scale (similar to those described in the previous chapter in the case of the RF-ACE algorithm and Google Compute), but as is often the case in machine learning, their re-invention occurs through proliferation, changes in scale, re-distributions of knowledge and infrastructure and specific optimisations. While machine learners in their

8. In any case, social media and search engines cannot be understood apart from the machine learning techniques that have been thoroughly woven through them since their inception. Hence *Elements of Statistical Learning* devotes several pages to Google's famous *PageRank* algorithm, describing it as an unsupervised learner (Hastie, Tibshirani, and Friedman 2009, 576-578).

machine form can be assigned a privileged position in the transformations of knowledge and action today, human machine learners are not exactly marginalized, at least in high profile cases such as Ng, Le Cun, Hinton and others. Rather, the scale of machine learning seems to be changing both for the algorithms and for the human computer scientists, programmers and engineers.

What accounts for this acceleration and slowing-down, the intensified interactions and abandonments of neural nets over the last three decades? Despite their apparent differences in origin, neural net share much with other machine learners. The language of brain, neurones and cognition associated with neural net covers over their much more familiar vector space and function-finding optimisations they rely on. Diagrammatic groupings and lines of movement operate in neural nets to expand their architecture in alignment with a series of existing machine learners. ‘The central idea,’ write Hastie and co-authors, ‘is to extract linear combinations of the inputs as derived features, and then model the target as a nonlinear function of these features. The result is a powerful learning method, with widespread applications in many fields’ (Hastie, Tibshirani, and Friedman 2009, 389). The ‘central idea’ can be seen in the algebraic expressions that Hastie and co-authors provide for the basic neural net model:

$$\begin{aligned} Z_m &= \sigma(\alpha_0 m + \alpha_m^T X) m = 1, \dots, M \\ T_k &= \beta_0 k + \beta_k^T Z, k = 1, \dots, K, \\ f_k(X) &= g_k(T), k = 1, \dots, K, \end{aligned} \tag{8.1}$$

where $Z = (Z_1, Z_2, \dots, Z_M)$, and $T = (T_1, T_2, \dots, T_K)$. The activation function $\sigma(v)$ is usually chosen to be the sigmoid $\sigma(v) = 1/(1 + e - v)$

(392)

```

function!sigmoid
machine learner!logistic
regression
cost function
diagram!equation as
linear model

```

Equation 8.1 is a diagram with some familiar elements as well as some novelty. Some of the diagrammatic operation of the neural net is already familiar from the linear models. The neural networks traverse data in a vector space denoted by X . That is common to nearly all machine learners.

They make use of the non-linear sigmoid function that lies at the heart of one of the main linear classifiers used in machine learning, logistic regression. Their training and learning processes have come to rely on the same kinds of cost, loss or error functions we have seen in other machine learners.

Their apparently increasing power to learn (to see, to find, to classify, to rank, to predict) owes much to the diagrammatic movements that recombine operations of past machine learners in new intersections. These movements appear in the equations. Equation 8.1 has three lines rather than one, and this layering and its diagonal patterns of indexical referencing running between subscripts distinguishes neural nets from the linear models it assembles.

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^p X_j \hat{\beta}_j \quad (8.2)$$

Whereas the standard linear model shown in Equation 8.2 indexes a single common vector space X_j and approximates a single function \hat{Y} by searching for the values of the parameters β_j that best incline a plane through the given data, the three lines shown in equation 8.1 are woven through each other much more consecutively. Much hinges on the unobtrusive sigmoid function operator written as σ : ‘a neural network can be thought of as a nonlinear generalization of the linear model, both for regression and classification. By introducing the nonlinear transformation σ , it greatly enlarges the class of linear models’ (394). σ , it seems, allows neural nets to

dataset!Titanic generalize beyond the linear model.⁹

Kaggle.com

The common diagrammatic operations of neural nets and other supervised machine learners immediately appears in almost any actual example of a neural net. In the code vignette shown below, the data is a spreadsheet of information about passengers of the Titanic. The `titanic` dataset, like `iris` or `boston` is often used in contemporary machine learning pedagogy. It is for instance, the main training dataset used by (kaggle.com)[<http://kaggle.com>], an online machine learning competition site I will discuss below . The first few lines of the R code load the dataset and transform it into vector space. For instance, variables such as `sex` that take values such as `male` and `female` become vectors of 1 and 0 in a new variable `sexmale`.

```
library(neuralnet)
titanic = read.csv("data/titanic3.csv")
titanic_transformed = as.data.frame(model.matrix(~survived + age + pclass +
                                               fare + sibsp + sex + parch + embarked, titanic))
train_index = sample.int(nrow(titanic)/2)
titanic_train = titanic_transformed[train_index, ]
titanic_net = neuralnet(survived ~ age + pclass + fare + sexmale + sibsp +
                        embarkedC + embarkedQ + embarkedS, data = titanic_train, err.fct = "ce",
                        linear.output = FALSE, rep = 5, hidden = 3, stepmax = 10000)
titanic_test = titanic_transformed[-train_index, ]
test_error = round(sum(0.5 < compute(titanic_net, titanic_test[, -c(1, 2)])) / 2)

## Error in nrow[w] * ncol[w]: non-numeric argument to binary operator
```

The line of the code that constructs a neural net using the `neuralnet`

9. Recent work on deep belief networks replaces the sigmoid function with other non-linear functions that subtly alter the way layers of neural nets relate to each other. See (Glorot and Bengio 2010) for an account of changing training practices in neural nets.

library (Fritsch and Guenther 2012), and the description of the classifier here is a familiar one. Despite its biological inspiration, the R formula for the neural net looks very similar to other machine learners such as logistic regression. It models whether someone **survived** the wreck of the Titanic in terms of their age, class of fare (**pclass**), sex, number of siblings/spouse (**sibsp**), number of parents/children (**parch**) and port of departure:

```
survived ~ age + pclass + fare + sexmale + sibsp + parch +
embarkedC + embarkedQ + embarkedS
```

R model formula express the response or target variable **survived** as a combination of other variables. In this case, the plus sign + indicates that the combination is linear or additive. As Hastie puts, ‘the central idea is to extract linear combinations of the inputs’ or predictor variables. If this model formula looks so similar to other machine learning techniques we have been discussing, what do neural networks add? Why did and do so many people turn to them?

In the code vignette above, the expression **hidden = 5** points to the distinctive architecture of these models, an architecture that does not appear in the model formula in the R code but does, as we have already seen, operate in the lines of Equation 8.1. The hidden units are indexed in the first line of the model in the variables Z_m . These ‘hidden units’ are key to neural net since they construct the ‘derived features’ that the model learns from the input data X . As Rumelhart, Winton and Williams announce the algorithm in a letter to *Nature* in 1986 entitled ‘Learning representations by back-propagating errors’ :

We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to

minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal ‘hidden’ units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure (Rumelhart, Hinton, and Williams 1986, 533).

Again, despite the persistent reference to biology, the description of the ‘new learning procedure’ starts to sound more like machine learning. There is talk of minimizing a measure of difference between actual and desire output vectors (optimizing through a cost or loss function), as well as mention of ‘features’ and ‘weights’ (usually a synonym for model parameters: ‘the neural network model has unknown parameters, often called weights, and we seek values for them that make the model fit the training data well’ (Hastie, Tibshirani, and Friedman 2009, 395)). The novelty, however, consists in the ‘hidden’ units whose interactions ‘represent important features.’ In other words, the flat additive combination of features expressed in the R model formula above does not convey the interactions of these units. As Hastie and co-authors put it, ‘the units in the middle of the network, computing the derived features Z_m , are called hidden units because the values Z_m are not directly observed’ (393). These units can only viably interact in the neural nets because the back-propagation algorithm offers a way to create ‘useful, new features’ from the data. But because they interact through back-propagation, the hidden units ‘capture’ regularities in the ‘task domain’ and thereby do what counts as cognition in the connectionist philosophies associated with neural nets (see the PDP group’s work (McClelland and

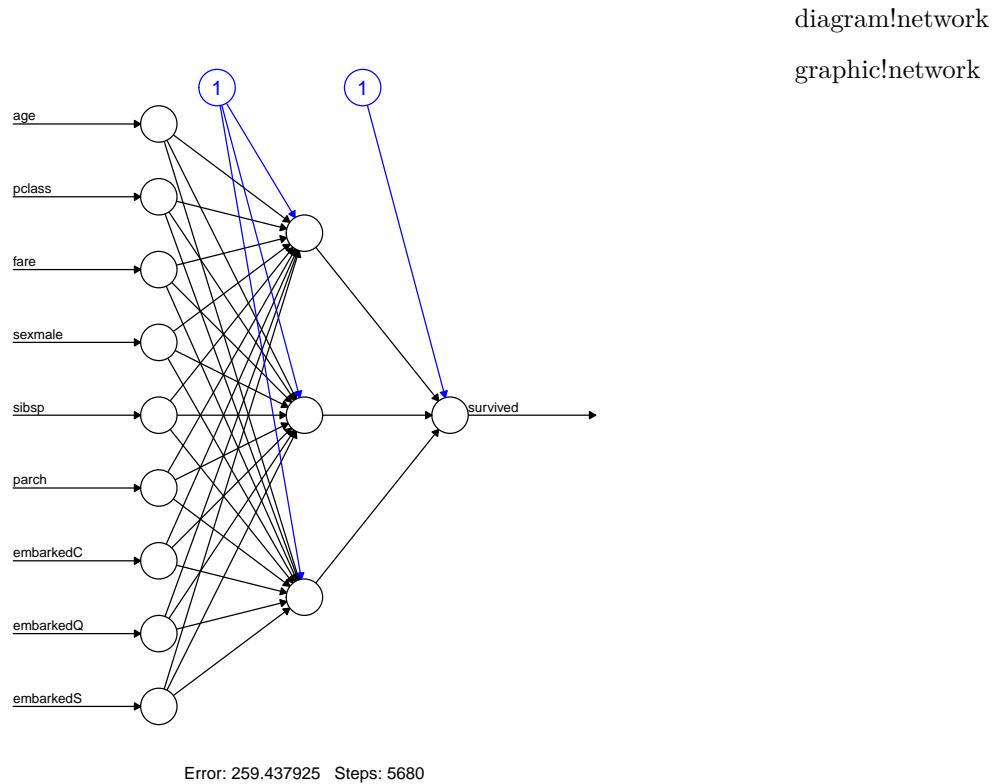


Figure 8.2: Neural network topology for 5-hidden unit ‘titanic’ data

Rumelhart 1986)).

The final major form in which neural net appear is the network diagram. Network graphs already appeared in Rosenblatt’s perceptron work (Rosenblatt 1958), but they ramify tremendously in the aftermath of back-propagation. Almost every book and article relating to neural net presents some version of the diagram shown in Figure 8.2.

Although the network topology of the model appears in many more complicated forms, it does several things in neural net literature. First, it presents a surface – the input layer – that indexes something in the world. The input layer, shown as X in the algebraic diagram of equation 8.1, acts as an organ, an eye or perhaps a camera. Early neural net papers on the handwritten digital recognition problem sometimes describe cameras mounted above

algorithm!back-propagation tables focused on images (LeCun et al. 1989). Second, it presents an output machine learner!neural layer that can contain single or multiple nodes, the k of equation 8.1. In the net!hidden nodes *titanic* examples, a single target node appears (survived or not). In the MNIST handwritten digit recognition models, there are usually ten output nodes, one for each of the digits 0 ... 9. Third, the network diagram orders forms of movement. Data and calculation propagate from bottom to top or vice-versa. (Sometimes the networks are rotated, and the flow is horizontal, but still bi-directional). Bi-directional hierarchical movement is key to the back-propagation algorithm in feed-forward and more complicated recurrent and convolutional neural nets. Fourth, it renders visible in principle the vital hidden nodes. Without the hidden nodes, neural nets revert to linear models. With the hidden nodes, the Z_m of the equations 8.1, neural nets, like some other machine learners we have discussed such as support vector machines, effectively expand the common vector space by constructing new dimensions in it. The derived features or ‘learned representations’ (to use the language of (Rumelhart, Hinton, and Williams 1986)) can expand indefinitely, according to different network topologies. Hidden nodes and hidden layers can multiply, bringing many new interactions and relations into the model (as we see in more recent revivals of neural net as deep learning (Hinton and Salakhutdinov 2006)).

The subjects of a hidden operation

Given the diagrammatic forms of the basic model equations, the network diagram and the operational code comprising the privileged machine at work recognising handwritten digits or classifying the passengers on the Titanic, how are subject positions assigned? The layered architecture of model generates new textures, densities, pools, sparsities and saturations in both the common vector space, and in the operations of machine learners.

How would we describe the figure of human machine learner in this setting? machine learner!kittydar
 Is the human machine learner like Vlad, the former Eastern European mathematician tending the neural net at the heart of the call centre knowledge management system, or more like Heather Arthur, the programmer who wrote kittydar? In *The Archaeology of Knowledge*, Michel Foucault refers to the ‘position of the subject’ as an anchor point for the power-laden, epistemically conditioning enunciative functions called ‘statements.’ When a subject position can be assigned, propositions, diagrams, numbers, models, calculations and data structures can come together in statements, as ‘the specific forms of an accumulation’ (Foucault 1972, 125). But this anchor point is not a unifying point grounded in interiority, in intentionality or even in single speaking position or voice (that of *the* machine learning expert, for instance). On the contrary, ‘various enunciative modalities manifest his [sic] dispersion’ (54). In the mist of this dispersion, the position of subject derive from operations that determine statements that become a kind of law for the subject.

As Foucault puts it, in a formulation that anticipates accounts of performativity that gain currency elsewhere several decades later,

in each case the position of the subject is linked to the existence of an operation that is both determined and present; in each case, the subject of the statement is also the subject of the operation (he who establishes the definition of a straight line is also he who states it; he who posits the existence of a finite series is also, and at the same time, he who states it) ; and in each case, the subject links, by means of this operation and the statement in which it is embodied, his future statements and operations (as an enunciating subject, he accepts this statement as his own law) (94-95).

subject
 position!back-propagation
 machine learner!subject
 as|sees subject position
 machine learner!Net-5

Foucault's examples here include subjects who say things like 'I call straight any series of points that ...': just such statements operate in machine learning. The *operation* is crucial, since it connects many different practices and techniques (function finding, optimisation, transformation of data into the common vector space, mobilisation of probability distributions as a kind of rule of existence for learnable situations, etc.) that accompany, ornament, armour and diagram the statement. Foucault posits a subject-positioning circularity between the operation and accompanying statement: the subject of the statement is also the subject of the operation. The provisional coincidence of operation and statement both creates a subject position, with some agency and subjects machine learners to future operations. The process might be formalised for any machine learner as follows: the diagrammatic operations of the machine learner support the production of statements; these operations become a way of producing future statements to the extent that the subject of the operation is *also* the subject of the statement. The assignation of a subject position occurs in this forward and backward, feed-forwarding and back-propagating movement between operation and statement.

Although the gap between operation and statement might seem small, there are many slippages and divergences in it. A minor statements such as 'we see that Net-5 does the best, having errors of only 1.6%, compared to 13% for the "vanilla" network Net-2' (Hastie, Tibshirani, and Friedman 2009, 407) bears within it, in its coupling to all the operations comprising 'Net-5,' a set of determinations and relations for variously positioned subjects. (These might include machine learners, such as Hinton or Le Cun, but also U.S. Postal workers, whose work changes as automatic mail sorting improves). In any concrete situation, in relation to any specific machine learner, the diagrammatic operations and statements will position subjects in specific

ways. There is no simple referent here, no simple object facing a knowing algorithm!back-propagation or controlling subject, since on this account, the operations and statements error in their dispersions, accumulations and distributions overflow any simple dyadic relation between a subject-object or human-machine/world.

Algorithms that propagate errors

The distinctive feature of neural nets, at least in their ordinary ‘vanilla’ forms, consists in their use of gradient descent to minimise errors by adjusting the weights (or parameters) of all the nodes (or linear models) comprising the machine learner. Adjusting the parameters of the nodes in the neural net hardly seems a striking achievement. If we, however, look more closely at the way in which the ‘interesting internal representations’ (Rumelhart, Hinton, and Williams 1986, 536) are iteratively constructed in neural nets, something more interesting begins to emerge from the forwards and backwards movement of this algorithm. Could an algorithm such as back-propagation diagram the slippery coincidence of subject of operation and subject of statement that defines machine learners?

The zone of slippage between statement and operation appears as error. Error rates, training error, test error, generalization error, validation error: these are just some of the errors that criss-cross between human and machine learners. Errors move between operations and statements. While not all of these errors figure directly in the algorithms, the learning procedure of most machine learners derives from the way they update model parameters in the light of incoming data, and errors or differences between expected and actual outputs. Every machine learner makes different determinations in relation to model parameters and errors. We have already seen something of the forward movement. It is defined by the equations 8.1 that move data

through a succession of layers and their nodes. Conversely, the equations 8.3 draw out how the back-propagating phase of a neural net update the weights of various nodes in the output layer nodes and the hidden layer nodes during each iteration:

$$\begin{aligned}\beta_{km}^{(r+1)} &= \beta_{km}^{(r)} - \gamma_r \sum_{i=1}^N \frac{\partial R_i}{\partial \beta_{km}^{(r)}} \\ \alpha_{ml}^{(r+1)} &= \alpha_{km}^{(r)} - \gamma_r \sum_{i=1}^N \frac{\partial R_i}{\partial \alpha_{km}^{(r)}}\end{aligned}\tag{8.3}$$

(Hastie, Tibshirani, and Friedman 2009, 396)

Many different variables figure in the equations 8.3. They include measures of error (R), values of the weights or parameters in various layers of the models (β, α), variables that count the number of iterations the model has performed ($r, r+1$) and the functional operators such as summation (Σ) and partial differentiation (∂). The usual indexical relations to data appear in N , the number of rows or observations, as well as K , the number of outputs and M , the number of nodes in the hidden layer. In the densely iconic and indexical diagram of equation 8.3, the interweaving of the subscripts in the two lines show how values of the model parameters of the first two lines of equation 8.1 are varied as the model is trained on the input data. The two lines of equation 8.3 specify how first the values of the parameters of the K nodes of the output layer should be altered in the light of the difference between the actual and expected output values, and then how the weight of the M nodes of the hidden layers should be adjusted. Once these are adjusted, the forward movement defined by equations 8.1 begins again. In adjusting weights in the layers, back-propagation always starts at the outputs, and travels back into the net towards the input layer at the bottom (or left hand side in diagram 8.2. ‘It is as if the error propagates from

the output y back to the inputs and hence the name *back-propagation* was coined' writes Alpaydin (Alpaydin 2010, 250). As in any gradient descent operation (see chapter 4), a rate parameter (here γ) regulates the speed of descent. If γ is too large, the gradient descent might jump over a valley that contains the absolute minimum error; if γ is too small, then the descent is too slow for fast machine learning. In some versions of neural net, the value of γ changes each at iteration r of the model.¹⁰

gradient descent
machine learning!competition!errors
in
machine learning!competition!as
examination

Competitions as examination

A related feed-forward and back-propagation of errors positions the machine learner subject, the 'wonderful people' of Hilary Mason's exhortation to developers. At almost every step of its development as a field and in almost every aspect of its operation, competitions to reduce error rates bring human and machine learners together. In competition, errors are not purely epistemic. They circulate within a wider economy of competitive optimisation that connect them to power, value and agency. The learning of machine learning takes place in examinations that rank both human and non-human machine learners according to error rates. What can we learn

10. If back-propagation was formulated in the 1980s (and indeed, was already known in 1960), what do we learn from its current re-iterations? Given the effort that went into crafting neural nets to recognise handwritten digits during the 1980s and 1990s, what does the revival of neural nets suggest about machine learning as feed-forward/back-propagation operation? From the early publications such as (Rumelhart, Hinton, and Williams 1985) on, the layered composition of the model has been linked to architectural considerations. As Hastie and co-authors write:

The advantages of back-propagation are its simple, local nature. In the back propagation algorithm, each hidden unit passes and receives information only to and from units that share a connection. Hence it can be implemented efficiently on a parallel architecture computer (Hastie, Tibshirani, and Friedman 2009, 397).

These practical considerations have different significance in different settings. Some of the current iterations of neural nets in deep learning rely on massively parallel computing architectures (for instance, Andrew Ng's GoogleX Youtube video project). Yet the information sharing that happens during back-propagation might also encompass the human others of neural nets. The efficient parallel implementation in computing architecture affects, I would suggest, human and non-human machine learners in different ways.

Kaggle|seemachine learning!competition!Kaggle from such competitions about subject positions in machine learning?

Something of the backwards and forwards movement between human and machine machine learners characterises competitions run by [Kaggle](#). Kaggle organizationally implements a parallel architecture machine learning process by back-propagating errors to hidden nodes embodied in individual competitors who, in principle at least, are not connected to each other, but only to the layers and nodes of Kaggle itself as a platform. In comparison to the research-oriented machine learning competitions such as the annual (KDD Cup)[<http://www.sigkdd.org/kddcup/index.php>] run by the Association of Computing Machinery (ACM) Special Interest Group on Knowledge Discovery and Data Mining, the NIPS (Neural Information Processing Systems) Challenges, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC [2014](#)) or the International Conference on Machine Learning (ICML)[<http://machinelearning.org/icml.html>], the Kaggle competitions attract a wide range of academic, industry, commercial and individual entries. Competitors no doubt enter these competitions for various reasons, not the least of which is their employment prospects or the promotion of their machine learning products (for instance, the winner of a major competition, the Heritage Health Fund Prize in 2012 uses that prize to promote the data mining software made by his company (Tiberius)[Tiberius.biz]; an entrant in the Hewlett ‘Automated Essay Competition’ again in 2012 included Pacific Metrics, a US company whose automated essay scoring products were already in use in U.S. schools; while Pacific Metrics did not win the competition, it acquired the winning machine learner and incorporated it into its products ([Kaggle 2012](#))). Kaggle.com is effectively a recruitment agency for machine learners ([Kaggle 2015c](#)). Certain competitions have recruitment as the prize (for instance, several competitions sponsored by Facebook have positions as data scientist at Facebook as the prize).

Ever wonder what it's like to work at Facebook? Facebook and Kaggle are launching an Engineering competition for 2015. Trail blaze your way to the top of the leader board to earn an opportunity at interviewing for a role as a software engineer, working on world class Machine Learning problems (Kaggle 2015b)

cross-validation
data diversity
common vector space

While most employment agencies rely on CVs (curriculum vitae), Kaggle employs something more like back-propagation and then cross-validation between multiple competitions as a way of optimising its ranking and prospective employment of machine learners.

In Figure 8.3, the competition organizers list three injunctions: download (the data), build (a model), and submit (an entry or many entries to the competition). Leader-boards and individual rankings within the Kaggle's "world's largest community of data scientists" (Kaggle 2015a),¹¹ allow clients of Kaggle (corporations mostly) to 'harness the "cognitive surplus"' (Kaggle 2015e). The figure also shows some of the typical diversity of the several hundred machine learning competitions that Kaggle has staged since 2011: diabetic retinopathy and west Nile virus prediction competition appear next to search results relevance or context ad clicks competitions. This proximity, whether accidental or constructed, between very disparate entities suggests that machine learners possess epistemic mobility not readily available to the domain experts in diabetes, virology, information retrieval or search engine optimisation. And again, the re-framing of their differences in a common vector space, subject to classification and optimisation is standard is a standard feature.

Like a neural network with many layers and nodes, Kaggle subjects the several hundred thousand competitors, like nearly all the machine learners

11. At the time of writing, Kaggle claims around 320,000 competitors.

error!generalization

The screenshot shows the Kaggle homepage. At the top, there's a blue header with the text "Welcome to Kaggle's data science competitions.", "New to Data Science? Tutorials on the Titanic competition.", "Want to learn from other's code? Kaggle's top rated scripts.", and three main buttons: "Download" (with a magnifying glass icon), "Build" (with a gear icon), and "Submit" (with a star icon). Below the header, there's a section titled "Active Competitions" with a button "All Competitions" (which is highlighted in blue) and a sub-section showing "15 found, 15 active". A search bar "Search competitions" is also present. To the right, there's a table listing three competitions:

Competition Name	Reward	Teams	Deadline
Diabetic Retinopathy Detection Identify signs of diabetic retinopathy in eye images	\$100,000	394	54 days
West Nile Virus Prediction Predict West Nile virus in mosquitoes across the city of Chicago	\$40,000	1035	14 days
Search Results Relevance Predict the relevance of search results from eCommerce sites	\$20,000	607	33 days

Figure 8.3: Kaggle data science competitions

we have been discussing, to ranking and indeed prediction based on the generalization error of the models that they submit to the competition. The leader-board, which displays current rankings of competitors in a given competition, is the visual form of this error-based ranking. The ranking and classification of machine learners, however, possibly goes much deeper than this:

The leaderboard is a central fixture of the Kaggle experience. It provides context to the incredible work accomplished by the Kaggle data science community. To a competitor, the leader-board is a dynamic, living, action-filled battle. Tactics come to life. Individuals leapfrog over each other. Teams merge and blend submissions. Some submit early and often, attempting to build up insurmountable leads. Others bide time, waiting to pounce minutes before the buzzer with their finest of forests. We see the joys of regularization and the agony of overfitting. It's raw. It's beautiful. It's thousands of hours of collective human toil (Kaggle 2015e)

The dynamics of ranking, and the experience of being ranked here arise

from a fairly simple mechanism. Entrants in a given competition download two datasets, a training dataset that includes labels for all the response variables, and a test dataset that does not include the labels. In principle, competitors construct machine learners using the training dataset, use their machine learner to predict labels for the test dataset, and then upload the predicted labels to Kaggle as a submission to the competition. The Kaggle platform then calculates a ranking based on the generalization error in the test labels. Kaggle itself has the actual labels for the test dataset. Entrants monitor the leaderboard and attempt to improve their rankings by making new submissions with improved or altered models. The many entries that participants sometimes submit to the competitions suggest that rankings, and their visibility operate like the loss functions that many supervised machine learners use to optimise the fit of a model to the training data. Competitors optimise their entries against each other, but the competition overall functions as a kind of general optimization process in which many hidden nodes adjust their treatment to the training data as scores and rankings propagate through via the leaderboard system. The very stylised injunction to download-model-submit many times effectively creates an algorithmic process in which many hidden nodes operate in parallel to produce predictions.

It would be possible to explore in much greater ethnographic depth the practices of Kaggle competitors, the spectrum of participants (ranging from undergraduate student teams through to retired scientists, from hedge fund financial analysts to physicists), and the ways in which the topics of competitions relate to different scientific, governmental and commercial problems. Here I am interested mainly in the form of the competition as a test or examination centred on errors. The competitions all take the form of examinations that set a problem, define some limits or constraints on its

error!generalization
 subject
 position!examination
 as normalizing

solution, and create a space that qualifies, ranks and displays the work of individuals or groups according to rates of generalization error (the error that arises when a machine learner encounters new data).

Machine learning competitions furnish a contemporary instance of the practices of examination that Foucault described in *Discipline and Punish*:

The examination combines the techniques of an observing hierarchy and those of a normalizing judgement. It is a normalizing gaze, a surveillance that makes it possible to qualify, to classify and to punish. It establishes over individuals a visibility through which one differentiates them and judges them. That is why, in all the mechanisms of discipline, the examination is highly ritualized. In it are combined the ceremony of power and the form of the experiment, the deployment of force and the establishment of truth. At the heart of the procedures of discipline, it manifests the subjection of those who are perceived as objects and the objectification of those who are subjected. The superimposition of the power relations and knowledge relations assumes in the examination all its visible brilliance (Foucault 1977, 183-185).

The disciplinary form of the examination of errors and errancy links statements and operations. Examinations combine ceremony, ritual, experiment, force and truth, as well as processes of subjectification and objectification. If, as suggested above, machine learning coalesced around the competitive handcrafting of neural nets during the 1990s, we might say that the consolidation of machine learning as a data practice today in competitions occurs via a much more pervasive practice of examining and testing. How would such a process be legible? The forms of visibility created by competitions

individualize and normalize machine learners (often by proper names), and maximise extractions of force, time, propensities and aptitudes. Machine learning competitions effectively implement a parallel architecture for machine learners with limited communication. The leaderboard ranks efforts to drive down error rates through collective testings, the same efforts that perhaps first affected machine learning around handwritten digit recognition in the 1990s. Entrants work hard to optimise the generalization errors of their models, but Kaggle aligns their work in a bootstrap aggregating that assigns subject positions according to error rates.

In many Kaggle competitions (some titles are shown in table 8.2), winning entries come from machine learners working together. In the National Data Science Bowl competition of 2015, the winning team comprised seven graduate and post-doctoral researchers from Ghent University, Belgium. In a jointly written blog account of their winning entry, team ‘Deep Sea’ describe something of the construction of the deep learning models they built. These were convolutional neural nets, neural nets in which unit of the network only ‘look’ at overlapping tiles of the input images:

We started with a fairly shallow models by modern standards (~ 6 layers) and gradually added more layers when we noticed it improved performance (it usually did). Near the end of the competition, we were training models with up to 16 layers. The challenge, as always, was balancing improved performance with increased overfitting (Dieleman 2015).

Like many of the entrants in image-based classification competitions such as the ImageNet Large Scale Visual Recognition Challenge (ILSVRC 2014), ‘Deep Sea’ built their machine learner in several stages, first deriving features from the data by creating various layers that looked for common features

learning, supervised

across various scales, rotations and other transformations of the plankton images, and then adding neural net layers to classify those derived features using the labels supplied in the training set. In this respect, and in almost perfect synchrony with the deep learning teams at Google, Facebook and many other places, ‘Deep Sea’ combined supervised and unsupervised learning techniques . The lower convolutional layers that process the images are strictly speaker unsupervised because they make no use of the known labels or categories of the plankton; the upper layers are supervised because they make use of the labels in the normal back-propagation process of neural net training.

In comparison to the neural nets discussed above, the deep belief networks involve many more parameters, stages of observation and modelling, configuration of hardware and infrastructural arrangements and comparison of results. ‘Deep Sea’ describe the architecture of one of their more successful models:

It has 13 layers with parameters (10 convolutional, 3 fully connected) and 4 spatial pooling layers. The input shape is (32, 1, 95, 95), in bc01 order (batch size, number of channels, height, width). The output shape is (32, 121). For a given input, the network outputs 121 probabilities that sum to 1, one for each class.

They go on to describe the different layers – cyclic slice, convolutional, spatial pooling – that derive features from the data or augmenting it - by examining overlapping tiles, by rotating or scaling the images, so that any given image, is ‘seen’ in a number of different ways, and the model learns to detect these variations. The combination of diverse layers in a stratified model introduces many different parameters into the model

and implies heavy computational work. Crucially, the massive parallel computing allows ‘deep’ learning. Infrastructure and cognition entwine heavily here, since the very possibility of building many layered neural nets depends on the possibility of relatively quickly training them. Probably many other competitors in this competition would not have had access to the Tesla K40 or ‘NVIDIA GTX 980 Superclocked’ GPU cards that ‘Deep Sea’ relied on.¹² Even with that intensive computational resource, their models required ‘between 24 and 48 hours to reach convergence.’ They constructed around 300 models. Because of the plethora of models with different architectures and parameters, ‘we had to select how many and which models to use in the final blend. For this, we used cross-validation on our validation set’ (Dieleman 2015). As is often the case, competitive examination engenders populations of machine learners whose aggregate tendencies stage model – in the sense of optimum – performance. The competition itself, then, creates of a visible hierarchy amidst a population of machine learners. Both during and after the competition, the visibility of the leaderboard gives access to and elicits further efforts to render visible, usually in the form of descriptions of winning entries and sometimes positing of model code.¹³ The ‘DeepSea’ team might be a typical intersectional entity today. Like the ‘wonderful people’ described by Hilary Mason, they bring together infrastructure, engineering, mathematics/statistics and some knowledge of human behaviour (although the knowledge of human behaviour

12. As another competitor in the National Data Science Bowl mentions:

One example is here the Kaggle plankton detection competition. At first I thought about entering the competition as I might have a huge advantage through my 4 GPU system. I reasoned I might be able to train a very large convolutional net in a very short time – one thing that others cannot do because they lack the hardware (Dettmers 2015)

Hardware parallelism and vectorization, at least in the area of deep learning, seems to matter more than the ability to test, examine, observe or invest new model configurations.

13. On the command line, `git clone https://github.com/benanne/kaggle-ndsb` makes a copy of the model code. The code in that github repository gives some idea of the mosaic of techniques, configurations, variations and tests undertaken by ‘DeepSea.’

feature engineering
statements!and
operations!zone of
slippage

Hinton, Geoffrey
machine learner!neural
net!mind of

in this case might have more to do with what other Kaggle competitors might be doing, as well as an awareness of the cutting edge research leaders in the field of visual object recognition techniques).

Subject positions in the confusion matrix

The members of ‘DeepSea’ built models that classify more than a hundred kinds of plankton with few errors. But in driving down error rates, they ineluctably occupy the subject position assigned by the conjunction of operation and the statements in machine learning. Neural nets in the form of deep belief nets are already automating through their many convolutional layers the feature engineering that characterised the skilled configuration work in machine learning in decision trees, linear regressions, support vector machines and predecessor neural nets. In all of this augmentation, pre-processing, unsupervised and supervised training, the competitions assign subject positions to machine learners amidst science, industry, media and government. The subject position of a machine learner, however, exposes its occupants to the zone of diagrammatic slippage between statements and operations. Neural nets strongly diagram the act of seeing machine learning itself. Hinton’s references to ‘mind’ might make sense here as part of the re-drawing human-machine differences. If some machine learners, such as neural nets, afford visibility to what they see in the world, then their function as ‘minds,’ as cognitive processes – I think this is what Hinton has in mind as a cognitive psychologist – effectively does begin to internalise the operations and statements that previously were handled by human machine learners. This back-propagating re-distribution privileges the machine as site of learning, but also quickly assigns new subject positions.

The growth of neural nets figures the vicissitudes of machine learners. As

the models grow, they test the capacity of human machine learners to understand how models relate to data. They test their knowledge and experience of how to craft and regularize the parameters of models in a given situation. Perhaps more profoundly, the growth of neural nets figure the deeply competitive imperative that frames much of the practice in machine learning, and in many ways constrains thinking around machine learning. This competition is not always explicit or overt but it almost transpires in the form of a test or examination. On behalf of machine learners, Andrew Ng declares that ‘we care about generalization error,’ . Error is neither the responsibility simply of the machine or its human others. Rather, error moves forwards and backwards between human and non-human machine learners.

Neural nets re-iteratively draw human-machine learning differences. Their own ups and downs, their potential to drive down error or learn features from data given enough data derives less from some exotic mathematical abstraction or encompassing algorithm, and more from an accumulation of layers and connections between units of modelling. The central algorithm – back-propagation – is instructive here: because it propagates errors throughout all the elements of the network, and every element in the network adjusts its weights in trying to minimise error, the layers can grow, becoming convolutional and recurrent on many scales. The predictive power of the model derives from the networked collective of linear models driven to optimise their error rates. So too, the competitive examinations that today generalize machine learning as a data practice predicate the ongoing potential of hidden units – machine learners – to learn from their rankings in tests of error.

The back-propagation of errors, understood as a technical figure for a subject position, is deeply affective. The cultural theorist Lauren Berlant describes

Ng, Andrew
error!generalization
examination
feature
back-propagation!and
growth of neural nets
subject
position!technical
figure of

machine learning!affect optimism as an operation:
in!optimism

The surrender to the return to the scene where the object hovers in its potentialities is the operation of optimism as an affective form (Berlant 2007, 20)

Berlant's complicated formulation brings together surrender, return, scene, object, potentialities and affective form. These movements, places and things might be understood as purely psychic or semiotic processes. But optimism as an asignifying diagrammatic operation also plays out across the manifold surfaces of algorithms, datasets, models, platforms and ranking systems associated with machine learning as a competitive examination. The 'object hovers in its potentialities' in the case of machine learning because neural nets and their kin assimilate and adjust their weights in response to changes in infrastructures and in the generalization of operations to newly adjacent domains. Machine learners generate optimism through and about optimisation, an optimisation that is deeply predictive, prospective, anticipatory and abductive. But this adjusting of weights carried out through the propagation of errors is also inherently a ranking or examination.

Human and machine machine learner differences can be re-drawn in two different directions. In one direction, machine learning operations assign a subject position. Neural nets epitomise this operation. The subjects who operate the neural net in order to fit a model find themselves deeply caught up in a network of neural nets that feed forward into a widening stream of parallel and layered architectures and operations. This feeding-forward, however, is regularized or narrowed down through examination and error, through back-propagation on various scales that ranks and filters machine learners according to their error rates. In this direction, the practice of

training and testing generalization error that has long guided the supervision of machine learners becomes a mechanisms for adjusting the positions of human machine learners. Some will be wonderful people, some will remain remote like Vlad, and some will optimistically re-learn in order to change their ranking.

Statistics	Count		Interdisci-	Count		Computer	Sci-	Count	
			plinary					Engineering	
1202	em	229		ma-	8318		ma-	4505	
	al-			chine			chine		
	go-			learn-			learn-		
	rithm			ing			ing		
579	vari-	205		em	6966		data	3553	
	able			al-			min-		
	se-			go-			ing		
	lec-			rithm					
	tion								
469	lasso	162		clas-	3869		clas-	2334	
				sifi-			sifi-		
				ca-			ca-		
				tion			tion		
423	clas-	118		vari-	3460		sup-	2102	
	sifi-			able			port		
	ca-			se-			vec-		
	tion			lec-			tor		
				tion			ma-		
							chine		
409	ma-	112		data	2549		clus-	1852	
	chine			min-			ter-		
	learn-			ing			ing		
	ing								
284	model	91		sup-	2440		neu-	1435	
	se-			port			ral		
	lec-			vec-			net-		
	tion			tor			work		
				ma-					
				chine					

Competition	Re-	domain	data_type
	ward_amount		
Heritage Health Prize Identify patients who will be admitted to a hospital within the next year using historical claims data Enter by 06 59 59 UTC Oct 4 2012	500000	health	measurements
GE Flight Quest Think you can change the future of flight	250000	transport	events
Flight Quest 2 Flight Optimization Milestone Phase Optimize flight routes based on current weather and traffic	250000	traffic	events
Flight Quest 2 Flight Optimization Main Phase Optimize flight routes based on current weather and traffic	220000	traffic	measurements
Flight Quest 2 Flight Optimization Final Phase Final Phase of Flight Quest 2	220000	traffic	measurements
National Data Science Bowl Predict ocean health one plankton at a time	175000	science	images
The Hewlett Foundation Automated Essay Scoring Develop an automated scoring algorithm for student written essays	100000	education	texts
The Hewlett Foundation Short Answer Scoring Develop a scoring algorithm for student written short answer responses	100000	education	texts
GE Hospital Quest Think it's possible to make hospital visits hassle free GE does	100000	health	actions
Diabetic Retinopathy Detection Identify signs of diabetic retinopathy in eye images	100000	medicine	images
Allstate Purchase Prediction Challenge Predict a purchased policy based on transaction history	50000	retail	transaction
Merck Molecular Activity Challenge Help de-	40000	medicine	measurements

Chapter 9

Conclusion: Out of the Data

If a new kind of operational reality is broadly coming into view through formations such as machine learning, formations in which people and things, knowledge and power, recombine in novel forms, then understanding the distribution of elements that make up this emerging common space of decision, classification, prediction and anticipation matters vitally. In the closing pages of *The Archaeology of Knowledge*, Foucault writes:

the positivities that I have tried to establish must not be understood as a set of determinations imposed from the outside on the thought of individuals, or inhabiting it from the inside, in advance as it were; they constitute rather the set of conditions in accordance with which a practice is exercised, in accordance with which that practice gives rise to partially or totally new statements, and in accordance with which it can be modified. These positivities are no so much limitations imposed on the initiative of subjects as the field in which that initiative is articulated (Foucault 1972, 208-209).

Here Foucault refers to the restricted freedom that discursive practices and formations open for us. It is increasingly difficult for science, media, government and business to think and act outside data. The generalization of machine learning frames statements and makes things visible today. And yet Foucault is quite clear that amidst the positivities of knowledge production, knowing the conditions, setting out the rules, and identifying the relations that striate the density and complexity of practice is a pre-condition to any transformations in practice.

I've read articles and books, downloaded data and software libraries, watched Youtube lectures and presentations, configured and written bits of code and text, made plots and diagrams, and done much configuration work across various platforms (Github.com, linux, Google Compute, R, python and ipython). Amidst all of this data practice, there is no reason to assume that learning machine learning is something that a person does as a conscious subject. When we look at an equation, when we try to comply with the machine learning injunction 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), the 'learning' may not be entirely that of a proper human subject. The material-semiotic dynamic does not readily map onto the forms of subjectivity, sense-making and experience that we typically take as anchor points for our life in the world.

I see writing about that learning as a practice of diagrammatically mapping the re-iterative drawing of human-machine relations. Moving into the data like or as a machine learner perhaps allows writing to become more diagrammatic. 'Between the figure and the text we must admit a whole series of crisscrossings' wrote Foucault (Foucault 1972, 66). For present purposes, I have been treating datasets as a kind of texts and the machine learning

as the crisscrossing that move data into various figures, moving the data diagrammatic!writing into various forms of visibility and across epistemological, infrastructural, referential and experiential thresholds.

After a couple of hundred weeks spending time with machine learners, some of which time has been mired in technical articles or the mathematics entries in Wikipedia, some of which has lost in a kind of dazzled and maybe naive immersion in technical configuration work, and some of which has been attempting to look to such techniques as ways of re-figuring contemporary forms of knowledge, power, value and experience, it seems to me that machine learning is an uneasy mixture of massively repeated and familiar forms, and something that is not easily understood. On the one hand, the level of imitation, duplications, copying and reproduction associated with the techniques suggests that a process of remaking the world according to particular forms is in process (for instance, in chapter 5 we saw how Naive Bayes classifiers are almost demonstrated on spam classification problems.) The scientific and engineering literature, with its really frequent variations on similar themes, suggests that imitation and copying are very much at the heart of the movements I have been describing. This is nothing new. It would be strange if these techniques were not subject to imitation and emulation. That imitation is predictable, we expect it and can account for it sociologically.¹ Some symptoms of these imitative fluxes can be found in the scientific and engineering literature. As we have seen, work on image and video classification, on text and speech, on gene interaction prediction or above all, on predictions of relations or associations between people and things (usually commodities, but not always) is striking in its persevering homogeneity. Moreover, the powerful aspirations evident amongst large

1. Accounts that might do this can be found in science and technology studies, particularly in actor-network theory versions, as well as in recent social and cultural theory that, for instance, draws on the work of the 19th century French sociologist, Gabriele Tarde (Tarde 1902; Borch 2005).

Husserl, Edmund!on
surface thing-shape

media platforms such as Baidu, Google and Facebook to re-ground machine learning in the project of artificial intelligence amidst social media or web page-related data in many ways continues business as usual for computer scientists (Gulcehre 2014).

At a deeper level, the techniques often engage with the world in somewhat predictable ways. Their intuitions of shape are largely governed by a set of fairly basic practical forms. The phenomenologist, Edmund Husserl, describes something similar in *The Origin of Geometry*:

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 attempts to describe something of the way in which forms such as circles, triangles, squares, lines, and points became objects of geometrical practice, but a similar polishing and smoothing of surfaces is certainly taking place today in the thing-shapes we call data.² The strong alignment to lines and smooth surfaces plays out in many of the optimisation techniques, and in the graphs of machine learning performance. Whatever else is happening in the functional mappings traced out in the algorithms, lines, curves and

2. In the essay on the origin of geometry, Husserl goes on to develop an account of how geometry opens up the possibility of a universality and infinite progress in knowledge constitutive of European civilization. Much of this sounds very problematic today, even if practically speaking the expansion of the styles of thought and practice Husserl was describing seems to confirm his account. The problem, to state it as plainly as possible, is that hardly anyone today thinks that geometry is a universal, not least because of the proliferation of mathematical thought and its production of alternative geometries.

surfaces bring with them powerful and predictable order to the operation of machine learning!unpredictable operation of the techniques.

As a data practice, however, machine learning is not entirely predictable. Machine learners, as we have seen, vary too much, they are biased, they overfit, they underfit, or they fail to generalise. The meta-techniques of cross-validation, bagging or ROC curves attempt to restore order. New machine learners arise from diagrammatic superimposition of existing practices or procedures. Neural networks are like a massively proliferating nest of perceptrons. Moreover, machine learning techniques often repeat something familiar by very different means (think of how **kittydar** treats photographs, or how a decision tree is legible but often unfamiliar). The event, then, resides less in either something intrinsic to devices operating as algorithmic models, or in something about the domains and places in which the devices operate (biomedicine, state security and intelligence agencies, finance, business, commerce, science, etc.). Perhaps it is a rather more modest event in which the tending of abstractions through estimation, optimisation, high-dimensional vectorisation, probabilistic mixing of latent and feature variables, and imputation unevenly replace existing ontological and epistemic norms of verification, objectification, and attribution.

I have been less interested in treating these techniques as the predictable re-animation of alienated reason, and more inclined to look for those elements in machine learning that diagrammatically abstract away from structures of representations, subjectification or indeed physicalization associated with platforms, services and products (for instance, the interminable implementations of document classifiers, sentiment analyses, or image labelling, or handwritten digit recognition, or autonomous navigation, etc.). Like Anne-Marie Mol's 'praxiography,' which seeks to maintain reality multiples in describing practice (Mol 2003, 6), the description of machine learning as

Mol, Anne-Marie!on
praxiography
data practice!as multiple

data practice intends to sustain the multiple of reality by identifying the practices that make it multiple. If a book could be a machine learner, then this one might be a generative model that seeks to maximise the range of practices it observes around the data. This reality maximisation might be entropic rather than convergent. It might not result in infinitely expanded $N = \text{all}$ -type infrastructures subsuming all data. In such a model, a model that would learn machine learning in order to intensify the abstraction , the features would figure less as techniques than as diagrams and diagrammatic process.

Bibliography

- Abramowitz, Milton. 1965. *Handbook of mathematical functions : with formulas, graphs, and mathematical tables*. In collaboration with Irene A. Stegun and United States. National Bureau of Standards. Dover Books.
- Ackley, David H., Geoffrey E. Hinton, and Terrence J. Sejnowski. 1985. “A learning algorithm for Boltzmann machines.” *Cognitive science* 9 (1): 147–169.
- ACM. 2013. “John M. Chambers - Award Winner.” Awards: Software System. Accessed December 12, 2013. http://awards.acm.org/award_winners/chambers_6640862.cfm.
- Adams, Vincanne, Michelle Murphy, and Adele E Clarke. 2009. “Anticipation: Technoscience, life, affect, temporality.” *Subjectivity* 28 (1): 246–265.
- Adler, Joseph, and Jörg Beyer. 2010. *R in a Nutshell*. O'Reilly Germany.
- Agency, National Security. 2012. “SKYNET: Courier Detection via Machine Learning.” The Intercept. Accessed October 29, 2015. <https://theintercept.com/document/2015/05/08/skynet-courier/>.
- Alpaydin, Ethem. 2010. *Introduction to machine learning*. Cambridge, Massachusetts; London: MIT Press.

- Amoore, Louise. 2011. "Data Derivatives On the Emergence of a Security Risk Calculus for Our Times." *Theory, Culture & Society* 28 (6): 24–43.
- Anthony, Sebastian. 2012. "Google Compute Engine: For \$2 million/day, your company can run the third fastest supercomputer in the world | ExtremeTech." ExtremeTech. Accessed October 3, 2012. [http://www.extremetech.com/extreme/131962-google-compute-engine-for-2-millionday-your-company-can-run-the-third-fastest-supercomputer-in-the-world](http://www.extremetech.com/extreme/131962-google-compute-engine-for-2-million-day-your-company-can-run-the-third-fastest-supercomputer-in-the-world).
- Aristotle. 1975. *Aristotle's Categories and de Interpretatione*. Translated by J.L. Ackrill. Oxford University Press.
- . 1981. *Nicomachean Ethics*. Hammondsorth, UK: Penguin Books.
- Arthur, Charles. 2015. "Artificial intelligence: don't fear AI. It's already on your phone – and useful." The Guardian. June 15. Accessed July 9, 2015. <http://www.theguardian.com/technology/2015/jun/15/artificial-intelligence-ai-smartphones-machine-learning>.
- Arthur, Heather. 2012. "harthur/kittydar." GitHub. Accessed September 16, 2014. <https://github.com/harthur/kittydar>.
- Bailey, N. T. J. 1965. *Probability methods of diagnosis based on small samples*. London: HM Stationery Office, London.
- Barad, Karen. 2007. *Meeting the universe halfway: Quantum physics and the entanglement of matter and meaning*. Duke University Press Books.
- Barber, David. 2011. *Bayesian reasoning and machine learning*. Cambridge; New York: Cambridge University Press.

- Baracas, Solon, Sophie Hood, and Malte Ziewitz. 2013. *Governing Algorithms: A Provocation Piece*. SSRN SCHOLARLY PAPER ID 2245322. Rochester, NY: Social Science Research Network.
- BBC. 2012. “Google ‘brain’ machine spots cats.” *BBC News: Technology* (June 26).
- Beer, David, and Roger Burrows. 2013. “Popular culture, digital archives and the new social life of data.” *Theory, Culture & Society*.
- Bellman, Richard. 1961. *Adaptive control processes: a guided tour*. Vol. 4. Princeton university press Princeton.
- Beniger, James R. 1986. *The control revolution: technological and economic origins of the information society*. Harvard University Press.
- Beniger, James R., and Dorothy L. Robyn. 1978. “Quantitative Graphics in Statistics: A Brief History.” *The American Statistician* 32, no. 1 (February 1): 1–11.
- Berlant, L. 2007. “Nearly utopian, nearly normal: Post-Fordist affect in La Promesse and Rosetta.” *Public Culture* 19 (2): 273.
- Bertin, Jacques. 1983. *Semiology of Graphics: Diagrams, Networks, Maps*. Madison: University of Wisconsin Press.
- Bishop, Christopher M., et al. 1995. *Neural networks for pattern recognition*. Cambridge; New York: Cambridge University Press.
- Bishop, Christopher M. 2006. *Pattern recognition and machine learning*. Vol. 1. New York: Springer.
- Blei, David M., and John D. Lafferty. 2007. “A correlated topic model of science.” *The Annals of Applied Statistics*: 17–35. JSTOR: [4537420](#).

- Blei, David M., Andrew Y. Ng, and Michael I. Jordan. 2003. “Latent dirichlet allocation.” *the Journal of machine Learning research* 3:993–1022.
- Bogost, Ian. 2012. *Alien phenomenology, or what it’s like to be a thing*. U of Minnesota Press.
- Bollas, Christopher. 2008. *The evocative object world*. London & New York: Routledge.
- Borch, Christian. 2005. “Urban Imitations: Tarde’s Sociology Revisited.” *Theory Culture Society* 22 (3): 81–100.
- Boyd, Stephen, and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge ; New York: Cambridge university press.
- Breiman, L. 2001. “Random forests.” *Machine learning* 45 (1): 5–32.
- Breiman, Leo. 2001. “Statistical modeling: The two cultures (with comments and a rejoinder by the author).” *Statistical Science* 16 (3): 199–231.
- Breiman, Leo, Jerome Friedman, Richard Olshen, Charles Stone, D. Steinberg, and P. Colla. 1984. “CART: Classification and regression trees.” *Wadsworth: Belmont, CA* 156.
- Butler, Declan. 2013. “When Google got flu wrong.” *Nature* 494, no. 7436 (February 13): 155–156.
- Campbell-Kelly, Martin. 2003. *From Airline Reservations to Sonic the Hedgehog: A History of the Software Industry*. Cambridge, MA: MIT Press.
- Cassirer, Ernst. 1923. *Substance and Function*. Translated by William Curtis Swabey and Marie Curtis Swabey. Chicago: Open Court Publishing.

- Chen, Xi, and Hemant Ishwaran. 2012. “Random forests for genomic data analysis.” *Genomics* 99, no. 6 (June): 323–329.
- Cheney-Lippold, John. 2011. “A new algorithmic identity soft biopolitics and the modulation of control.” *Theory, Culture & Society* 28 (6): 164–181.
- Church, Alonzo. 1936. “A note on the Entscheidungsproblem.” *Journal of Symbolic Logic* 1 (1): 40–41.
- . 1996. *Introduction to mathematical logic*. Princeton N.J.: Princeton University Press.
- Cleveland, William S., Eric Grosse, and William M. Shyu. 1992. “Local regression models.” *Statistical models in S*: 309–376.
- CNN. 2011. “40 Under 40: Ones to watch.” CNNMoney. Accessed January 22, 2013. http://money.cnn.com/galleries/2011/news/companies/1110/gallery_40_under_40_ones_to_watch.fortune/.
- Coleman, Gabriella. 2012. *Coding freedom: the ethics and aesthetics of hacking*. Princeton N.J.: Princeton University Press.
- Collins, Harry M. 1990. *Artificial experts: Social knowledge and intelligent machines*. Inside technology. Cambridge, MA.: MIT Press.
- Conway, Drew, and John Myles White. 2012. *Machine learning for hackers*. Sebastopol, CA: O’Reilly.
- Cortes, C., and V. Vapnik. 1995. “Support-Vector Networks.” *Machine Learning* 20, no. 3 (September): 273–297.
- Couldry, Nick. 2012. *Media, society, world: Social theory and digital media practice*. Cambridge ; Malden, MA: Polity.

- Cover, Thomas, and Peter Hart. 1967. "Nearest neighbor pattern classification." *Information Theory, IEEE Transactions on* 13 (1): 21–27.
- Cox, Geoff. 2012. *Speaking Code: Coding as Aesthetic and Political Expression*. MIT Press.
- Cramer, J. S. 2004. "The early origins of the logit model." *Studies in History and Philosophy of Science Part C: Studies in History and Philosophy of Biological and Biomedical Sciences* 35 (4): 613–626.
- CRAN. 2010. "The Comprehensive R Archive Network." Accessed May 5, 2010. <http://www.stats.bris.ac.uk/R/>.
- Cranor, Lorrie Faith, and Brian A. LaMacchia. 1998. "Spam!" *Communications of the ACM* 41 (8): 74–83.
- Dahl, George. 2013. "Deep Learning How I Did It: Merck 1st place interview." No free hunch. Accessed June 17, 2013. <http://blog.kaggle.com/2012/11/01/deep-learning-how-i-did-it-merck-1st-place-interview/>.
- Deleuze, Gilles. 1988. *Foucault*. Translated by Seán Hand. Minneapolis: University of Minnesota Press.
- Deleuze, Gilles, and Félix Guattari. 1994. *What is philosophy?* Translated by Hugh Tomlinson. European perspectives. New York; Chichester: Columbia University Press.
- Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. "Maximum Likelihood from Incomplete Data via the EM Algorithm." *Journal of the Royal Statistical Society. Series B (Methodological)* 39, no. 1 (January 1): 1–38. JSTOR: 2984875.

Derrida, Jacques. 1989. *Edmund Husserl's Origin of geometry, an introduction*. Translated by John Leavey. Lincoln: University of Nebraska Press.

Dettmers, Tim. 2015. "Which GPU(s) to Get for Deep Learning: My Experience and Advice for Using GPUs in." Deep Learning. Accessed July 8, 2015. <https://timdettmers.wordpress.com/2014/08/14/which-gpu-for-deep-learning/>.

Dieleman, Sander. 2015. "Classifying plankton with deep neural networks." Sander Dieleman. Accessed July 3, 2015. <http://benanne.github.io/2015/03/17/plankton.html>.

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

Doyle, Peter. 1973. "The use of automatic interaction detector and similar search procedures." *Operational Research Quarterly*: 465–467. JSTOR: [10.2307/3008131](https://doi.org/10.2307/3008131).

Dreyfus, Hubert L. 1972. *What computers can't do*. New York: Harper & Row.

———. 1992. *What computers still can't do: a critique of artificial reason*. MIT Press.

Duda, Richard O., Peter E. Hart, and David G. Stork. 2012. *Pattern classification*. John Wiley & Sons.

Durbin, Richard, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. 1998. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. 1 edition. Cambridge, New York: Cambridge University Press, May 13.

- Edwards, Paul N. 1996. *The closed world : computers and the politics of discourse in Cold War*. Inside technology. Cambridge, Mass. ; London: MIT Press.
- Efron, B. 1979. "Bootstrap methods: another look at the jackknife." *The annals of Statistics* 7 (1): 1–26.
- Einhorn, Hillel J. 1972. "Alchemy in the Behavioral Sciences." *Public Opinion Quarterly* 36, no. 3 (September 21): 367–378.
- Ensmenger, Nathan. 2012. "Is Chess the Drosophila of Artificial Intelligence? A Social History of an Algorithm." *Social Studies of Science* 42, no. 1 (February 1): 5–30.
- Fico. 2015. "FICO® Analytic Modeler Decision Tree Professional | FICO™." FICO™ | FICO Decisions. Accessed November 1, 2015. <http://www.fico.com/en/products/fico-analytic-modeler-decision-tree-professional>.
- Fisher, R.A. 1938. "The statistical utilization of multiple measurements." *Annals of Human Genetics* 8 (4): 376–386.
- Fisher, Ronald A. 1936. "The use of multiple measurements in taxonomic problems." *Annals of eugenics* 7 (2): 179–188.
- Fix, Evelyn, and Joseph L. Hodges. 1951. *Discriminatory analysis-nonparametric discrimination: consistency properties*. DTIC Document.
- Flach, Peter. 2012. *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press.
- Foucault, Michel. 1972. *The archaeology of knowledge and the discourse on language*. Translated by Allan Sheridan-Smith. New York: Pantheon Books.

———. 1977. *Discipline and punish: The birth of the prison*. Translated by Allan Sheridan-Smith. New York: Vintage.

———. 1991. *The history of sexuality*. Translated by Robert Hurley. London: Penguin : Viking : Pantheon.

———. 1992 [1966]. *The Order of Things: An Archaeology of Human Sciences*. Translated by Allan Sheridan-Smith. London: Routledge.

———. 1998. *The Will to Knowledge: The History of Sexuality*. Translated by Robert Hurley. Vol. 1. London: Penguin.

Foucault, Michel, and Paul Rabinow. 1997. *Ethics: Subjectivity and Truth*. New York: New Press.

Friedman, Jerome H. 1997. "On bias, variance, 0/1—loss, and the curse-of-dimensionality." *Data mining and knowledge discovery* 1 (1): 55–77.

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

Galloway, A. R. 2014. "The Cybernetic Hypothesis." *differences* 25, no. 1 (January 1): 107–131.

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

- Garling, Caleb. 2015. "Andrew Ng: Why 'Deep Learning' Is a Mandate for Humans, Not Just Machines | WIRED." *Wired*. Accessed July 9, 2015. <http://www.wired.com/2015/05/andrew-ng-deep-learning-mandate-humans-not-just-machines/>.
- Gillespie, Tarleton. 2010. "The politics of 'platforms'" *New Media & Society* 12 (3): 347–364.
- . 2014. "9 The Relevance of Algorithms." *Media technologies: Essays on communication, materiality, and society*: 167.
- Gitelman, Lisa, ed. 2013. *"Raw Data" is an Oxymoron*. Cambridge, Massachusetts ; London, England: MIT Press.
- Glorot, Xavier, and Yoshua Bengio. 2010. "Understanding the difficulty of training deep feedforward neural networks." In *International conference on artificial intelligence and statistics*, 249–256.
- Gomes, Lee. 2014. "Machine-Learning Maestro Michael Jordan on the Delusions of Big Data and Other Huge Engineering Efforts - IEEE Spectrum." October 3. Accessed March 5, 2015. <http://spectrum.ieee.org/robotics/artificial-intelligence/machinelearning-maestro-michael-jordan-on-the-delusions-of-big-data-and-other-huge-engineering-efforts>.
- Gruber, John. 2004. "Markdown: Syntax." Daring Fireball: Markdown. Accessed July 1, 2013. <http://daringfireball.net/projects/markdown/>.
- Guattari, Félix. 1984. *Molecular revolution : psychiatry and politics*. Harmondsworth, Middlesex, England ; New York, N.Y., U.S.A.: Penguin.
- Guattari, Felix, and Gilles Deleuze. 1988. *A thousand plateaus: capitalism and schizophrenia*. London: Athlone, 1988.

- Gulcehre, Caglar. 2014. "Welcome to Deep Learning." *Deep Learning*. Accessed October 24, 2014. <http://deeplearning.net/>.
- Hacking, Ian. 1975. *The emergence of probability*. Cambridge ; New York: Cambridge University Press.
- . 1990. *The taming of chance*. Cambridge University Press.
- Hallinan, Blake, and Ted Striphas. 2014. "Recommended for you: The Netflix Prize and the production of algorithmic culture." *New Media & Society* (June 23): 1–21.
- Halpern, Orit. 2015. *Beautiful Data*. Durham, N.C: Duke University Press.
- Hand, D. J., and K. M. Yu. 2001. "Idiot's Bayes - Not so stupid after all?" *International Statistical Review* 69, no. 3 (December): 385–398.
- Haraway, Donna. 1997. *Modest-Witness@Second-Millennium.FemaleMan-Meets-OncoMouse: feminism and technoscience*. New York ; London: Routledge.
- Hardy, Quentin. 2010. "Power in the Numbers Page 2 of 3 - Forbes.com." Forbes.com. May 24. Accessed June 10, 2010. http://www.forbes.com/forbes/2010/0524/opinions-software-norman-nie-spss-ideas-opinions_2.html.
- Hastie, Trevor, Robert Tibshirani, and Jerome H. Friedman. 2001. *The elements of statistical learning: data mining, inference, and prediction*. 1st edition. New York: Springer.
- . 2009. *The elements of statistical learning: data mining, inference, and prediction*. 2nd edition. New York: Springer.

- Heis, Jeremy. 2014. “Ernst Cassirer’s Substanzbegriff und Funktionsbegriff.” *HOPOS: The Journal of the International Society for the History of Philosophy of Science* 4, no. 2 (September 1): 241–270. JSTOR: [10.1086/676959](https://doi.org/10.1086/676959).
- Helmreich, Stefan. 2000. *Silicon second nature : culturing artificial life in a digital world*. Berkeley, Calif. ; London: University of California Press.
- Henson, Joseph, German Tischler, and Zemin Ning. 2012. “Next-generation sequencing and large genome assemblies.” *Pharmacogenomics* 13, no. 8 (June): 901–915. pmid: [22676195](https://pubmed.ncbi.nlm.nih.gov/22676195/).
- Hey, T., S. Tansley, and K. Tolle, eds. 2009. *The fourth paradigm: data-intensive scientific discovery*. Microsoft Research.
- Hilary Mason - Machine Learning for Hackers*. 2012. June 6.
- Hinton, Geoffrey E. 1989. “Connectionist learning procedures.” *Artificial intelligence* 40 (1): 185–234.
- Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. 2006. “A fast learning algorithm for deep belief nets.” *Neural Computation* 18, no. 7 (July): 1527–1554.
- Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. 2006. “Reducing the dimensionality of data with neural networks.” *Science* 313 (5786): 504–507.
- Hof, Robert D. 2014. “Chinese Search Giant Baidu Thinks AI Pioneer Andrew Ng Can Help It Challenge Google and Become a Global Power.” MIT Technology Review. August 14. Accessed May 18, 2015. <http://www.technologyreview.com/featuredstory/530016/a-chinese-internet-giant-starts-to-dream/>.

- Hood, Leroy, and Daniel J. Kevles, eds. 1992. “Biology and medicine in the twenty-first century.” In *The Code of Code*, 136–63. Cambridge MA: Harvard University Press.
- Hothorn, Torsten. 2014. “CRAN Task View: Machine Learning & Statistical Learning” (December 18).
- Hothorn, Torsten, Kurt Hornik, and Achim Zeileis. 2006. “Unbiased Recursive Partitioning: A Conditional Inference Framework.” *Journal of Computational and Graphical Statistics* 15 (3): 651–674.
- IBM. 2013. “IBM PureData System.” Accessed June 21, 2013. <http://www-01.ibm.com/software/data/puredata/>.
- . 2014. “IBM’s Watson learns the language of science.” August 28. Accessed April 16, 2015. <https://www-03.ibm.com/press/us/en/pressrelease/44697.wss>.
- ILSVRC. 2014. “ImageNet Large Scale Visual Recognition Competition 2014 (ILSVRC2014).” Accessed July 6, 2015. <http://www.image-net.org/challenges/LSVRC/2014/>.
- Inc., Google. 2012. “Behind the Compute Engine demo at Google I/O 2012 Keynote - Google Compute Engine — Google Developers.” Accessed August 13, 2012. <https://developers.google.com/compute/io>.
- Issenberg, Sasha. 2012. “The Definitive Story of How President Obama Mined Voter Data to Win A Second Term | MIT Technology Review.” MIT Technology Review. Accessed January 9, 2013. <http://www.technologyreview.com/featuredstory/509026/how-obamas-team-used-big-data-to-rally-voters/>.

- James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An introduction to statistical learning*. Springer.
- Jockers, Matthew L. 2013. *Macroanalysis: Digital Methods and Literary History*. Urbana: University of Illinois Press.
- Kaggle. 2012. “The Hewlett Foundation: Automated Essay Scoring | Kaggle.” Accessed July 1, 2015. <https://www.kaggle.com/c/asap-aes>.
- . 2015a. “About | Kaggle.” Accessed June 3, 2015. <https://www.kaggle.com/about>.
- . 2015b. “Competitions | Kaggle.” Accessed June 3, 2015. <https://www.kaggle.com/solutions/competitions>.
- . 2015c. “Data Science Jobs Forum | Kaggle.” Accessed July 2, 2015. <https://www.kaggle.com/jobs>.
- . 2015d. “Description - Facebook Recruiting IV: Human or Robot? | Kaggle.” Accessed July 2, 2015. <https://www.kaggle.com/c/facebook-recruiting-iv-human-or-bot>.
- . 2015e. “Description - Leaping Leaderboard Leapfrogs | Kaggle.” Accessed June 4, 2015. <https://www.kaggle.com/c/leapfrogging-leaderboards>.
- KDD. 2013. “Call For KDD Cup.” Accessed July 23, 2013. <http://www.kdd.org/kdd2013/call-for-cup>.
- Keating, Peter, and Alberto Cambrosio. 2012. “Too many numbers: Microarrays in clinical cancer research.” *Studies in History and Philosophy of Science Part C: Studies in History and Philosophy of Biological and Biomedical Sciences* 43, no. 1 (March): 37–51.

- Khan, Javed, Jun S. Wei, Markus Ringner, Lao H. Saal, Marc Ladanyi, Frank Westermann, Frank Berthold, Manfred Schwab, Cristina R. Antonescu, Carsten Peterson, et al. 2001. "Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks." *Nature medicine* 7 (6): 673–679.
- Kirk, Matthew. 2014. *Thoughtful Machine Learning: A Test-Driven Approach*. 1 edition. Sebastopol, Calif.: O'Reilly Media.
- Kitchin, Rob. 2014. "Big Data, new epistemologies and paradigm shifts." *Big Data & Society* 1 (1): 2053951714528481.
- Klimt, Bryan, and Yiming Yang. 2004. "The enron corpus: A new dataset for email classification research." In *Machine learning: ECML 2004*, 217–226. Springer.
- Krzywinski, Martin I, Jacqueline E Schein, Inanc Birol, Joseph Connors, Randy Gascoyne, Doug Horsman, Steven J Jones, and Marco A Marra. 2009. "Circos: An information aesthetic for comparative genomics." *Genome Research*.
- Kuhn, Thomas S. 1996. *The structure of scientific revolutions*. Chicago, IL: University of Chicago Press.
- Lamport, Leslie, and A. LaTEX. 1986. *Document Preparation System*. Reading, MA: Addison-Wesley.
- Lander, Eric S., Lauren M. Linton, Bruce Birren, Chad Nusbaum, Michael C. Zody, Jennifer Baldwin, Keri Devon, et al. 2001. "Initial sequencing and analysis of the human genome." *Nature* 409, no. 6822 (February 15): 860–921.
- Lanier, Jaron. 2013. *Who owns the future?* London: Allen Lane.

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

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

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

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

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

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

Le, Quoc V., Marc’Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg S. Corrado, Jeff Dean, and Andrew Y. Ng. 2011. “Building high-level features using large scale unsupervised learning” (December 28). arXiv: [1112.6209](https://arxiv.org/abs/1112.6209).

Lecture 1 / Machine Learning (Stanford). 2008.

Lecture 10 / Machine Learning (Stanford). 2008. In collaboration with Andrew Ng. July 23.

Lecture 13 / Machine Learning (Stanford). 2008. In collaboration with Andrew Ng. July 23.

Lecture 2 / Machine Learning (Stanford). 2008. July 23.

Lecture 6 / Machine Learning (Stanford). 2008. In collaboration with Andrew Ng. July 23.

Lecture 7 / Machine Learning (Stanford). 2008. In collaboration with Andrew Ng. July 23.

Lecture 8 / Machine Learning (Stanford). 2008. In collaboration with Andrew Ng. July 23.

Lecture 9 / Machine Learning (Stanford). 2008. In collaboration with Andrew Ng. July 23.

LeCun, Yann, Bernhard Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne Hubbard, and Lawrence D. Jackel. 1989. “Backpropagation applied to handwritten zip code recognition.” *Neural computation* 1 (4): 541–551.

LeCun, Yann, and Corinna Cortes. 2012. “MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges.” Accessed June 24, 2013. <http://yann.lecun.com/exdb/mnist/>.

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

Leonelli, S. 2014. “What difference does quantity make? On the epistemology of Big Data in biology.” *Big Data & Society* 1 (1): 1–11.

- Lury, Celia, Luciana Parisi, and Tiziana Terranova. 2012. "Introduction: The Becoming Topological of Culture." *Theory, Culture & Society* 29 (4-5): 3–35.
- Lynch, Michael. 1993. *Scientific practice and ordinary action : ethnomethodology and social*. Cambridge: Cambridge University Press.
- Mackenzie, Adrian. 1997. "Undecidability: the history and time of the Universal Turing Machine." *Configurations* 3:359–379.
- . 2006. *Cutting code: software and sociality*. Digital Formations. New York: Peter Lang.
- . 2010. "Every thing thinks: Sub-representative differences in digital video codecs." In *Deleuze in Science and Technology Studies*, in collaboration with Caspar Bruun Jensen and Kjetle Rodje, 139–154. Oxford: Berghahn Publishers.
- . 2011. "More parts than elements: how databases multiply." *Environment and Planning D: Society and Space* 29 (6): 335–350.
- . 2012. "Sets." In *Devices and the Happening of the Social*, edited by Celia Lury and Nina Wakeford, 219–231. Routledge.
- . 2013a. "From Validating to Verifying: Public Appeals in Synthetic Biology." *Science as Culture* 22 (4): 476–496.
- . 2013b. "The economic principles of industrial synthetic biology: cosmogony, metabolism and commodities." *Engineering Studies* 5 (1): 74–89.
- . 2013c. "'Wonderful people': programmers in the regime of anticipation." *Subjectivity* 6 (4): 391–405.

- . 2014a. “Multiplying numbers differently: an epidemiology of contagious convolution.” *Distinktion: Scandinavian Journal of Social Theory* 15 (2): 189–207.
- . 2014b. “UseR! Aggression, Alterity and Unbound Affects in Statistical Programming.” In *Fun and Software: Exploring Pleasure, Paradox and Pain in Computing*, edited by Olga Goriunova. New York: Bloomsbury Academic.
- . 2015a. “Data.” Fieldsights - Theorizing the Contemporary, Cultural Anthropology Online, September 24. Accessed September 24, 2015. <http://culanth.org/fieldsights/712-data>.
- . 2015b. “Distributive numbers: a post-demographic perspective on probability.” In *Empirical Baroque*, edited by John Law and Evelyn Ruppert. Mattering Press.
- Mackenzie, Adrian, Matthew Fuller, Andrew Goffey, Mills , Richard, and Stuart Sharples. 2016. “Code repositories as expressions of urban life.” In *Code and the City*, edited by Rob Kitchin. London: Routledge.
- Mackenzie, Adrian, Richard Mills, Stuart Sharples, Matthew Fuller, and Andrew Goffey. 2015. “Digital Sociology in the Field of Devices.” In *Handbook of Sociology of the Arts and Culture*, edited by Mike Savage and Laurie Hanquinet. London & New York: Routledge.
- Mackenzie, Adrian, and Simon Monk. 2004. “From Cards to Code: How Extreme Programming Re-Embodies Programming as a Collective Practice.” *Computer Supported Cooperative Work (CSCW)* 13 (1): 91–117.

- Madrigal, Alexis C. 2014. "How Netflix Reverse Engineered Hollywood." The Atlantic. Accessed August 28, 2014. <http://www.theatlantic.com/technology/archive/2014/01/how-netflix-reverse-engineered-hollywood/282679/>.
- Malley, James D., Karen G. Malley, and Sinisa Pajevic. 2011. *Statistical Learning for Biomedical Data*. 1st ed. Cambridge University Press.
- Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. 1st ed. Cambridge University Press, July 7.
- Marchese, Dr Francis T. 2013. "Tables and Early Information Visualization." In *Knowledge Visualization Currents*, edited by Francis T. Marchese and Ebad Banissi, 35–61. Springer London, January 1.
- Markoff, John. 2012. "In a Big Network of Computers, Evidence of Machine Learning." *The New York Times* (June 25).
- Maron, M.E. 1961. "Automatic Indexing: An Experimental Inquiry." *Journal of the Association for Computing Machinery* 8:404–417.
- Marx, Karl. 1986. *Capital A Critique of Political Economy. The process of production of capital*. Moscow: Progress.
- Massumi, Brian. 2002. *Parables for the Virtual*. Durham, N.C: Duke University Press.
- . 2013. *Semblance and Event: Activist Philosophy and the Occurrent Arts*. Reprint edition. Cambridge, Mass.: MIT Press, September 6.
- Matloff, Norman S. 2011. "Art of R programming." Accessed June 26, 2013. <http://www.books24x7.com/marc.asp?bookid=44507>.

- Mayer-Schönberger, Viktor, and Kenneth Cukier. 2013. *Big Data: A Revolution that Will Transform how We Live, Work, and Think*. Boston: Eamon Dolan/Houghton Mifflin Harcourt.
- McClelland, James L., and David E. Rumelhart. 1986. *Parallel distributed processing. Explorations in the Microstructure of Cognition*. Vol. 1. Cambridge, MA & London: MIT Press.
- McCormack, Derek. 2012. "Geography and abstraction Towards an affirmative critique." *Progress in Human Geography* 36, no. 6 (December 1): 715–734.
- McKinney, Wes. 2012. *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. Sebastopol, CA: O'Reilly & Associates Inc.
- McKinsey. 2009. "Hal Varian on how the Web challenges managers -." McKinsey Quarterly - Strategy - Innovation. Accessed June 10, 2010. http://www.mckinseyquarterly.com/Hal_Varian_on_how_the_Web_challenges_managers_2286.
- McMillan, Robert. 2013. "How Google Retooled Android With Help From Your Brain." WIRED. February 18. Accessed August 4, 2015. <http://www.wired.com/2013/02/android-neural-network/>.
- McNally, Ruth, Adrian Mackenzie, Jennifer Tomomitsu, and Allison Hui. 2012. "Understanding the 'Intensive' in 'Data Intensive Research': Data Flows in Next Generation Sequencing and Environmental Networked Sensors." *International Journal of Digital Curation* 7 (1): 81–94.
- Meza, Juan C. 2010. "Steepest descent." *Wiley Interdisciplinary Reviews: Computational Statistics* 2 (6): 719–722.

- Minsky, Marvin, and Seymour Papert. 1969. "Perceptron: an introduction to computational geometry." *The MIT Press, Cambridge, expanded edition* 19:88.
- Mitchell, Tom M. 1997. *Machine learning*. New York, NY [u.a.: McGraw-Hill.
- Mohamed, Abdel-rahman, Tara N. Sainath, George Dahl, Bhuvana Ramabhadran, Geoffrey E. Hinton, and Michael A. Picheny. 2011. "Deep Belief Networks using discriminative features for phone recognition," 5060–5063. IEEE, May.
- Mohr, John W., and Petko Bogdanov. 2013. "Introduction—Topic models: What they are and why they matter." *Poetics* 41, no. 6 (December): 545–569.
- Mol, Annemarie. 2003. *The Body Multiple: Ontology in Medical Practice*. Durham, N.C: Duke University Press.
- Moore, David S. 2009. *The Basic Practice of Statistics*. 5th Edition. New York; London: W. H. Freeman.
- Morgan, James N., and John A. Sonquist. 1963. "Problems in the analysis of survey data, and a proposal." *Journal of the American Statistical Association* 58 (302): 415–434.
- Morton, Timothy. 2013. *Hyperobjects: Philosophy and Ecology After the End of the World*. Univ Of Minnesota Press.
- Muenchen, Robert A. 2014. "The Popularity of Data Analysis Software." R4stats.com. Accessed September 2, 2015. <http://r4stats.com/articles/popularity/>.

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

Myers, Eugene W., Granger G. Sutton, Art L. Delcher, Ian M. Dew, Dan P. Fasulo, Michael J. Flanigan, Saul A. Kravitz, Clark M. Mobarry, Knut HJ Reinert, Karin A. Remington, et al. 2000. “A whole-genome assembly of *Drosophila*.” *Science* 287 (5461): 2196–2204.

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

NIST. 2012. “Gallery of Distributions.” Engineering Statistics Handbook. Accessed September 21, 2012. <http://www.itl.nist.gov/div898/handbook/eda/section3/eda366.htm>.

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

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

Parry, R. M., W. Jones, T. H. Stokes, J. H. Phan, R. A. Moffitt, H. Fang, L. Shi, A. Oberthuer, M. Fischer, W. Tong, et al. 2010. “k-Nearest neighbor models for microarray gene expression analysis and clinical outcome prediction.” *The pharmacogenomics journal* 10 (4): 292–309.

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

———. 2015. “Anomaly Detection: The Mathematization of the Abnormal in the Metadata Society.” Berlin.

Pearson, Karl. 1901. "LIII. On lines and planes of closest fit to systems of points in space." *Philosophical Magazine Series 6* 2, no. 11 (November 1): 559–572.

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

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

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

Perez, Fernando, and Brian E. Granger. 2007. "IPython. A system for interactive scientific computing." *Computing in Science & Engineering* 9 (3): 21–29.

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

Pevzner, Pavel A., Haixu Tang, and Michael S. Waterman. 2001. "An Eulerian path approach to DNA fragment assembly." *Proceedings of the National Academy of Sciences* 98, no. 17 (August 14): 9748–9753. pmid: [11504945](#).

Quinlan, J. Ross. 1986. "Induction of decision trees." *Machine learning* 1 (1): 81–106.

Quinlan, John Ross. 1993. *C4. 5: programs for machine learning*. Vol. 1. San Francisco, Calif.: Morgan Kaufmann.

- R Development Core Team. 2010. “The R Project for Statistical Computing.” Accessed June 11, 2010. <http://www.r-project.org/>.
- Rabiner, Lawrence. 1989. “A tutorial on hidden Markov models and selected applications in speech recognition.” *Proceedings of the IEEE* 77 (2): 257–286.
- Rajaraman, Anand, and Jeffrey David Ullman. 2012. *Mining of massive datasets*. Cambridge University Press.
- Ramaswamy, Sridhar, Pablo Tamayo, Ryan Rifkin, Sayan Mukherjee, Chen-Hsiang Yeang, Michael Angelo, Christine Ladd, Michael Reich, Eva Latulippe, Jill P. Mesirov, et al. 2001. “Multiclass cancer diagnosis using tumor gene expression signatures.” *Proceedings of the National Academy of Sciences* 98 (26): 15149–15154.
- RexerAnalytics. 2010. “Rexer Analytics 4th Annual Data Miner Survey - 2010.” Accessed May 9, 2011. <http://www.rexeranalytics.com/Data-Miner-Survey-Results-2010.html>.
- Richert, Willi, and Luis Pedro Coelho. 2013. *Building Machine Learning Systems with Python*. Birmingham: Packt Publishing.
- Ripley, Brian. 1996. *Pattern recognition and neural networks. 1996*. Cambridge ; New York: Cambridge University Press.
- . 2014. *tree: Classification and regression trees*.
- Robinson, Derek. 2008. “Function.” In *Software studies: a lexicon*, edited by M. Fuller, 101–110. The MIT Press.
- Rose, N. 2009. “Normality and pathology in a biomedical age.” *Sociol. Rev.* 57:66–83.

- Rosenblatt, F. 1958. "The perceptron: A probabilistic model for information storage and organization in the brain." *Psychological Review* 65 (6): 386–408.
- Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. 1985. *Learning internal representations by error propagation*. DTIC Document.
- . 1986. "Learning representations by back-propagating errors." *Nature* 323, no. 6088 (October 9): 533–536.
- Russell, Matthew A. 2011. *Mining the social web*. Sebastopol, CA: O'Reilly.
- Savage, M. 2009. "Contemporary Sociology and the Challenge of Descriptive Assemblage." *European Journal of Social Theory* 12 (1): 155.
- Schutt, Rachel, and Cathy O'Neil. 2013. *Doing data science*. Sebastopol, Calif.: O'Reilly & Associates Inc.
- Segaran, Toby. 2007. *Programming collective intelligence: building smart web 2.0 applications*. Sebastopol CA.: O'Reilly.
- Shapin, Steven, and Simon Schaffer. 1989. *Leviathan and the air-pump : Hobbes, Boyle, and the experimental life*. Princeton ; Oxford: Princeton University Press.
- Slikker, W, Jr. 2010. "Of genomics and bioinformatics." *The pharmacogenomics journal* 10, no. 4 (August): 245–246. pmid: [20676063](#).
- Smith, David. 2012. "R analysis shows how UK health system could save £200m." Revolutions. Accessed December 12, 2013. <http://blog.revolutionanalytics.com/2012/12/nhs-prescription-analytics.html>.

- Smith, Marquard. 2013. "Theses on the Philosophy of History: The Work of Research in the Age of Digital Searchability and Distributability." *Journal of Visual Culture* 12, no. 3 (December 1): 375–403.
- Stamey, Thomas A., John N. Kabalin, John E. McNeal, Iain M. Johnstone, F. Freiha, Elise A. Redwine, and Norman Yang. 1989. "Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate. II. Radical prostatectomy treated patients." *The Journal of urology* 141 (5): 1076–1083.
- Steinberg, Dan, and Phillip Colla. 2009. "CART: classification and regression trees." *The Top Ten Algorithms in Data Mining*: 179–201.
- Stengers, Isabelle. 2005. "Deleuze and Guattari's Last Enigmatic Message." *Angelaki* 10 (1): 151–167.
- . 2008. "Experimenting with Refrains: Subjectivity and the Challenge of Escaping Modern Dualism." *Subjectivity* 22, no. 1 (May): 38–59.
- . 2011. *Cosmopolitics II*. Translated by Robert Bononno. University of Minnesota Press, September 26.
- Stevens, Hallam. 2011. "Coding Sequences: A History of Sequence Comparison Algorithms as a Scientific Instrument." *Perspectives on Science* 19 (3): 263–299.
- . 2013. *Life out of sequence: a data-driven history of bioinformatics*. Chicago, London: University Of Chicago Press.
- Stigler, Stephen M. 1986. *The history of statistics: the measurement of uncertainty before 1900*. Cambridge, Mass.: Harvard University Press.

- . 2002. *Statistics on the table: The history of statistical concepts and methods*. Harvard University Press.
- Stone, Mervyn. 1974. “Cross-validatory choice and assessment of statistical predictions.” *Journal of the Royal Statistical Society. Series B (Methodological)*: 111–147. JSTOR: [2984809](#).
- Suchman, Lucy. 2006. *Human and Machine Reconfigurations: Plans and Situated Actions*. 2nd ed. Cambridge University Press, December 4.
- Suchman, Lucy A. 1987. *Plans and situated actions : the problem of human-machine communication*. Cambridge: Cambridge University Press.
- Suchman, Lucy A, and Randall H Trigg. 1992. “Artificial intelligence as craftwork.” In *Understanding Practice: Perspectives on activity and context*, edited by Seth Chaiklin and Jean Lave, 144–178. Cambridge ; New York: Cambridge University Press.
- Sunder Rajan, Kaushik. 2006. *Biocapital : the constitution of postgenomic life*. Durham: Duke University Press.
- Tarde, Gabriel de. 1902. *Psychologie économique*. Paris, F. Alcan.
- Teator, Paul. 2011. *R cookbook*. O'Reilly Media, Incorporated.
- Thacker, Eugene. 2005. *The global genome : biotechnology, politics, and culture*. Cambridge, Mass.: MIT Press.
- Therneau, Terry, Beth Atkinson, and Brian Ripley. 2015. *rpart: Recursive Partitioning and Regression Trees*.
- Tibshirani, Robert. 1996. “Regression shrinkage and selection via the lasso.” *Journal of the Royal Statistical Society. Series B (Methodological)*: 267–288. JSTOR: [2346178](#).

- Totaro, Paolo, and Domenico Ninno. 2014. "The concept of algorithm as an interpretative key of modern rationality." *Theory, Culture & Society*: 29–49.
- Tuv, E., A. Borisov, G. Runger, and K. Torkkola. 2009. "Feature selection with ensembles, artificial variables, and redundancy elimination." *The Journal of Machine Learning Research* 10:1341–1366.
- Uprichard, Emma, Roger Burrows, and David Byrne. 2008. "SPSS as an 'inscription device': from causality to description?" *Sociological Review* 56 (4): 606–622.
- Valiant, Leslie G. 1984. "A theory of the learnable." *Communications of the ACM* 27 (11): 1134–1142.
- Van Dijck, José. 2012. "Facebook and the engineering of connectivity: A multi-layered approach to social media platforms." *Convergence: The International Journal of Research into New Media Technologies*: 1354856512457548.
- Vance, Ashlee. 2009. "Data Analysts Captivated by R's Power." *The New York Times: Technology / Business Computing* (January 7).
- . 2011. "This Tech Bubble Is Different." *BusinessWeek: magazine* (April 14).
- Vapnik, Vladimir. 1999. *The Nature of Statistical Learning Theory*. 2nd ed. 2000. Springer, December 1.
- Vapnik, Vladimir N., and A. Ya Chervonenkis. 1971. "On the uniform convergence of relative frequencies of events to their probabilities." *Theory of Probability & Its Applications* 16 (2): 264–280.

- Venables, William N., and Brian D. Ripley. 2002. *Modern applied statistics with S*. Springer.
- Venables, William, and B. D. Ripley. 2000. *S Programming*. Springer, April 20.
- Venter, J. Craig, Mark D. Adams, Eugene W. Myers, Peter W. Li, Richard J. Mural, Granger G. Sutton, Hamilton O. Smith, et al. 2001. “The Sequence of the Human Genome.” *Science* 291, no. 5507 (February 16): 1304–1351. pmid: [11181995](#).
- Virno, Paolo. 2004. *A grammar of the multitude : for an analysis of contemporary forms of life*. Semiotext(e) foreign agents series. Los Angeles: Semiotext(e).
- Warner, Homer R., Alan F. Toronto, L. George Veasey, and Robert Stephenson. 1961. “A mathematical approach to medical diagnosis: application to congenital heart disease.” *Jama* 177 (3): 177–183.
- Warner, Homer R., Alan F. Toronto, and L. George Veasy. 1964. “Experience with Baye’s Theorem for Computer Diagnosis of Congenital Heart Disease*.” *Annals of the New York Academy of Sciences* 115, no. 2 (July 1): 558–567.
- Wasserman, Larry. 2003. *All of statistics: a concise course in statistical inference*. New York: Springer.
- What we’re learning from online education / Video on TED.com*. 2012. In collaboration with Daphne Koller. August.
- Whitehead, Alfred North. 1956. *Modes of thought; six lectures delivered in Wellesley College, Massachusetts, and two lectures in the University of Chicago*. New York, Cambridge University Press.

- . 1960. *Process and reality, an essay in cosmology*. New York, Macmillan.
- Wikibooks. 2013. “R Programming - Wikibooks, open books for an open world.” Accessed June 27, 2013. http://en.wikibooks.org/wiki/R_Programming.
- Perceptron*. 2013. In *Wikipedia, the free encyclopedia*, by Wikipedia.
- Wilf, Eitan. 2013. “Toward an Anthropology of Computer-Mediated, Algorithmic Forms of Sociality.” *Current Anthropology* 54, no. 6 (December 1): 716–739. JSTOR: [10.1086/673321](https://doi.org/10.1086/673321).
- Wilson, Elizabeth A. 2010. *Affect and artificial intelligence*. University of Washington Press.
- Witten, Ian H., and Eibe Frank. 2005. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.
- Wu, X., V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J McLachlan, A. Ng, B. Liu, P. S Yu, et al. 2008. “Top 10 algorithms in data mining.” *Knowledge and Information Systems* 14 (1): 1–37.
- Xie, Yihui. 2013. “knitr: A general-purpose package for dynamic report generation in R.” *R package version* 1.
- Xie, Yihui, and J. J. Allaire. 2012. “New Tools for Reproducible Research with R.” Warwick, UK.
- Zare, Douglas. 2012. “Difference between logistic regression and neural networks - Cross Validated.” CrossValidated. December 7. Accessed May 26, 2015. <http://stats.stackexchange.com/questions/43538/difference-between-logistic-regression-and-neural-networks>.

- Zerbino, D. R., B. Paten, and D. Haussler. 2012. "Integrating Genomes." *Science* 336, no. 6078 (April 13): 179–182.

Index

- k-means clustering, 91

k-nearest neighbours model, 91

Elements of Statistical Learning, 32

{ , 92

‘kittydar’, 36

abstraction, 30–31

accounts of, 23

algorithm as, 12

diagrammatic, 180

equations as, 16

in algorithms, 11

levels of, 112

lived, 24

of line and plane, 208

operational practice of, 24

advertising, online, 15

algorithm

back-propagation, 262, 273, 276, 279

primacy, 10

recursive partitioning, 191, 194

recursivity, 11

Alpayadim, Ethem

on decision boundaries, 204

Amoore, Louise, 9

Anderson, Chris, 83

Aristotle

categories, 119

artificial intelligence, 3, 53, 54

rule-based induction, 190

automation

limits of, 12

back-propagation

and growth of neural nets, 291

see algorithm

back-propagation, 260

Badiou, Alain, 81

bagging, 75, 287

Barad, Karen

on experimental apparatus, 37

Bellman, Richard

curse of dimensionality, 84

biopower, 81

Bogost, Ian, 20, 59

- Bollas, Christopher, 65
- Breiman, Leo, 1, 74, 116
- CART monograph, 190
 - on statistical cultures, 151
 - on support vector machine, 209
- calculation
- affect of, 49
- calculus, 79
- differential, 96
- Cambrosio, Albert
- on microarrays, 229
- cancer, 77
- Cassirer, Ernst, 120
- categories, 14
- Chambers, John, 63
- Church, Alonzo, 99
- classification, 6, 85, 89
- algorithms for, 13
 - as ranking, 239
 - decision boundary, 201
- Cleveland, William, 4
- cloud computing, 99
- code
- brevity of, 192, 233
 - command line, 160
- coefficients, 96
- common vector space, 73, 89–90, 91, 94, 100, 100, 114, 124, oper-
- ations, 130, 283
- conventions of, 76
- dimension, 88, 91
- dimensionality, 101, 208
- curse of, 84, 245, 252
- epistemics, 264
- linear algebra, *see also* linear algebra
- metatable, 104
 - vector, 97
- vectorisation, 97, 102
- function, 100
- control, 8
- control revolution, 9
 - crisis of, 217
- Cortes, Corinna, 199, 268
- on pattern recognition, 179
- cost function, 271
- Couldry, Nick, 14
- Coursera, 47
- cross-validation, 283
- referentiality of, 230
- Cukier, Kenneth, 148
- data
- architecture
 - map-reduce, 73
 - assembly, 220
- dimensionality, 192

- DNA microarray, 217
form of
genomic, 220
latent variables in, 16
practice, 4
practices, 7
sequence, 251
strain, 73, 218
test, 115
training, 14, 115
variable
types, 85
variations, 244
volume, 72
wide, dirty, mixed, 231
data diversity, 283
data mining, 3
in 1970, 189
data practice
as multiple, 300
data science
relation to machine learning, 2
dataset, 76
iris, 75, 191, 194, 201, 231
MACQ-II, 244
MNIST, 202, 265
nci, 77
prostate, 86, 88, 93, 101
South African Heart Disease, 169
South African heart disease, 130
spam, 76, 77
spam email
Enron, 160
SRBCT, 232, 237, 265
Titanic, 272
U.S. Postal digits, *see* MNIST
datasets
diagrammatic character of, 234
decision, 10
decision tree, 17
deep learning, 93
Deleuze, Gilles, 29
calculus, 134
diagrams, 94
see also diagram
of power, 140
diagram, 29, 40–42, 51–52, 139
decision tree as, 197
diagrammaticism of the, 42
diagrammatization
as generalization, 223
equation as, 271
forms of movement, 207
hand-drawn, 46
icon, 41
indexical, 171
indexical sign, 41
machine learner as, 259

- mathematical function as, 139
- movement, 180
- network, 275
- of power, 140
- overlay, 261
- reference, 152
- statistical decomposition, 175
- world, 164
- diagrammatic
 - affect of, 49
 - diagonal, 113, 241
 - experiment, 121
 - substitution, 209
 - writing, 297
- diagrammatic movement, 83, 105
- diagrammatization, 51
- difference
 - Gini index of diversity, 193
- digital humanities, 16
- Domingos, Pedro, 29, 58, 109
- empiricities, 81
- enunciative modality, 197
 - seestatement
 - enunciative modality of, 181
- epistemologization
 - threshold of, *see also* epistopic
- epistemology, 83
- epistopic, 83
- error, 279
- generalization, 173, 283, 285, 291
- overfitting, 195
- variance, 195
- errors
 - bias-variance, 171
- examination, 291
- face recognition, 36
- facial recognition, 9
- feature, 291
 - selection, 287
- feature engineering, 290
- feature space, *see* common vector space
- Fisher, Ronald Ayre, 201
- Fix, Evelyn, 245
- Flach, Peter, 33
- Foucault, Michel, 8, 103, 257
 - disciplinary power, 238
 - on distribution, 154
 - on positivity, 182
 - on statements, 182
 - materiality of, 212
- Friedman, Jerome, 32
 - on bias-variance decomposition, 172
 - work on decision tree, 189
- function, 111

- as partial observer, 134, 139
biological, 219
cost, 196
cost or objective, 134
diagram, 123
diagrammatic operation of, 139
different senses of mathematical, algorithmic, operational, 112
discriminant, 201
enunciative of neural net, 260
kernel, 210
learning, 119–121
linear, 208
linear discriminant, 201
logistic, 124, 127
mathematical, 113, 116–118 invention of, 122
operational unit of code, 133
parameters of, 126
probability distribution Gaussian, 145, 155
probability distributions, 152
referentiality see also referentiality, 126
sigmoid, 271
transformation in meaning of,

Galloway, Alex, 17, 18
Gauss, Carl Friedrich, 89
generalization, 8, 75
genome as hyperobject, 224
genomes variations
single nucleotide polymorphism (SNP), 243
genomics
as cross-validation of machine learning, 224
importance in machine learning, 218
geometry
Husserl on universality of, 298
Google
Google Compute Engine, 225
Google Flu, 20
Google Trends, 3
I/O Conference, 2012, 224
gradient descent, 269, 281
graphic
Circo diagram, 225
heatmap, 77
line and curve, 122
network, 275

- graphics
 scatterplot matrix, 86
- Hacking, Ian
 on C.S. Peirce, 146
- Hammerbacher, Jeff, 15
- handwriting recognition, *see also*
 digit recognition
- Hastie, Jeff, 32
- Hinton, Geoffrey, 259, 261, 268, 290
- Husserl, Edmund
 on surface thing-shape, 298
- hyperobject, 33, 224
 genome as, 222
 machine learning as, 67
- hyperplane, *see* decision surface
- image recognition, 5
- infrastructure, 10
 reconfiguration of, 261
- Jockers, Matthew, 16
- k-means clustering, 6
- k-nearest neighbors, 4
- Kaggle.com, 272
- Keating, Paul
 on microarrays, 229
- Kirk, Matthew, 71, 110
- Kitchin, Rob
- on epistemology of 'big data', 182
- kittydar, 4, 256, 268
- Koller, Daphne, 32, 47
- Kuhn, Thomas, 38
- Lash, Scott
 on generative rules, 21
- Lazzarato, Maurizio
 asemiotic machine, 141
 machinic ontological mutation, 152
- Le Cun, Yann, 268
- learning, 48, 57
 as dividing, 193
- learning, supervised, 288
- linear algebra, 84, 92–95
- linear model, 91, 271
- linear regression, 4, 6, 44, 46, 91
- linear regression model, 39–40, 89
- Linnaeus, Carl, 79
- local regression, 4
- logistic function
 history of, 125
- Lynch, Mike, 83
- machine learner, 7
 $k\text{nearest neighbours}$, 157, 245
 $k\text{nearest neighbours}(KNN)$, 244
- automatic interaction detector,

- 186
C4.5, 190
CART, 189, 190
computer program as, 1
decision tree, 180
history of, 186
in medicine, 189
deep learning
existential threat of, 263
diagrammatic composition of, 12
epistopic mode, 84
function, 192
gender of, 258
hierarchical clustering, 218
k-nearest neighbours, 247
history, 246
kittydar, 211, 277
learning of, 135
Least Absolute Shrinkage and Selection Operator, 241
linear discriminant analysis, 6, 204
not applied to gene expression, 238
logistic regression, 6, 127, 271
mobility of, 104
Naive Bayes, 6, 156
history of, 169, 170
National Security Agency machine learning
Skynet, 2
Net-5, 278
neural net, 6, 180, 256
hidden nodes, 276
infrastructures of, 281
mind of, 290
optimisation of, 134
Ordinary Least Sum of Squares, 241
overfitting, 188
pattern recognition, *see also* pattern
perceptron, 48–50, 56–57, 262
probability distribution as control surface, 156
random forest, 74, 198
use in genomics, 236
RF-ACE, 227
SkyNet, 25
statistical decomposition of, 167
subject as, *see* subject position
support vector machine, 180
machine learners
Hidden Markov Model
in genome assembly, 221
ontological mutations of, 152
programs, 58
variety of, 109–111

- affect in
- optimism, 292
- agency of as mobility, 51
- as ordering practice, 15
- competition
 - as examination, 281
 - errors in, 281
 - competitions, 265
 - epistopic, 90, 105
 - error
 - bias-variance, 244
 - generative models in, 21
 - learning, 100, 114–116
 - positivity of, 250
 - probabilisation of, *see also* probabilisation
 - ranking of, 292
 - regularization, 14, 240
 - regularization in, 238
 - regularizing hyperobjects, 223
 - statistical aspects, 83
 - statistical practices, 147
 - structure differences, 151
 - structuring differences, 152
 - supervised, 185
 - unpredictable operation of, 299
- Maron, M.E., 170
- Marx, Karl, 62
- Mason, Hilary, 258
- Massumi, Brian, 92
- mathematics, 7
 - as stabilisation of practice, 12
- maximum likelihood
 - implementation of, 132
- Mayer-Schönberger, Viktor, 148
- medical diagnosis, 169
- Mitchell, Tom, 1, 33
- model
 - coefficient, 94
 - fitting, 73
 - generative, 21, 259
- Mohr, John, 16
- Mol, Anne-Marie, 59
 - on praxiography, 300
- Munster, Anna, 9
- natural language processing, 20
- Netflix, 20
- neural net, 51
- neural network
 - seemachine learner
 - neural net, 6
- Ng, Andrew, 21, 31, 44, 84, 258, 266, 268, 291
- Non-negative matrix factorization, 36
- ontology
- stochastic, 145

- operation, field of, 8
operations research
 use of decision trees, 188
ordinary least squares, 96
overfitting, 266
Parisi, Luciana, 92
Pasquinelli, Paolo, 11
pattern
 as a term in machine learning,
 180
 dispersion of, 201
 in dispersion, 198
 modes of togetherness, 179
 operational, 198
Pearson, Karl, 53
Peirce, C.S., 40
performativity, 277
phronesis, 38
 predictive, 19
positivity, 182, 219, 225, 295
 accumulation in, 183
 as basis of propositions, 228
 of knowledge, 7, 213
power, 14
practice, 59
 scientific, 83
prediction, 84
principal component analysis, 6
probabilisation
ancestral communities of, 167,
 170
probability
 conditional, 158
 distribution, 113
 history of, 165
programmability
problem of, 30, 266
 neural net as solution to, 262
programming
 functional, 99
 programming language
 FORTRAN, 31
 R, 59–65
 Comprehensive R Archive Net-
 work, 111
 S, 62
 programming languages, 60, 97
 as mode of writing, iv
 Python, v
 R, v
 Python, *see also* programming lan-
 guage
 scikit-learn library, 111
Quinlan, John Ross, 190
 induction tree, 190
R packages

- ElemStatLearn, 62
- MASS, 62
- random variable, 153
- referential, 197, 202
 - dispersed, 200
 - entanglement, 218
 - epistemologization
 - threshold of, 224
 - threshold of, 237
- things and activities, 76
- referentiality, 174, 176
- Ripley, Brian, 33, 66
- Rosenblatt, Frank, 48
- Savage, Mike, descriptive assemblage, 179
- science
 - experiment, 120
 - knowledge
 - positivity of, 232
- scientific publications, 23
- signal processing
 - relation to machine learning, 223
- similarity, 79
- social media platforms
 - machine learning as part of, 269
- source code, 59
- statement, 257
 - as diagram, 183
- enunciative function of, 184
- enunciative modality of, 184
- materiality of, 212
- position of subject, 255
- referential of, 197
- statements
 - and operations
 - zone of slippage, 290
- enunciative function, 277
- statistics, 74
 - biomedical
 - changes in, 229
 - history, 80
 - history of
 - from error to real quantity, 149
 - Law of Large Numbers, 219
 - probability distributions
 - normal, 149
 - tests, 96
- Stengers, Isabelle, 119
- subject position, 277
 - back-propagation, 278
 - examination as normalizing, 286
 - operational assignment of, 277
 - technical figure of, 291
- Suchman, Lucy, 258
 - on ancestral communities, 169
 - on machine-as-agent effect, 51
- support vector machine, 6

seemachine learner
support vector machine, 6

table, 67, 103, 104
history, 81
history , 78

Terranova, Tiziana, 92

thinking, 65

Tibshirani, Rob, 32

topic model, 16

topology, 92

Vapnik, Vladimir, 50, 117, 199
dimensional increase, 209

vector, 86

vectorisation, 72, 208
of infrastructure, 261

vectorization, 190
hardware, 289
infrastructural, 224

Venables, Bill, 66

Whitehead, A. North, 73

Whitehead, Alfred North
on pattern, 179

Wilson, Elizabeth
on artificial intelligence, 49