substantially different to any other project by virtue of the hybridization between code in text, and text in code. Practically, this is made possible by working on code and text within the same file, in the same text editor. Switching between writing `R` and Python code (about which I say more below) to retrieve data, to transform it, to produce graphics, to construct models or some kind of graphic image, and within the same file be writing academic prose, might be one way to write about machine learning as a data practice.

The capacity to mingle text, code and images depends on an ensemble of open source, often command-line software tools that differ somewhat from the typical social scientist or humanities researchers' software toolkit of word processor, bibliographic software, image editor and web browser. In particular, I have relied on software packages in the `R` programming language such as the '`knitr`' (Xie 2013; Xie and Allaire 2012) and in `python,` the `ipython` notebook environment (Perez and Granger 2007). Both have been developed by scientists and statisticians in the name of 'reproducible research.' Many examples of this form of writing can be found on the web: see IPython Notebook Viewer for a sample of these. These packages are designed to allow a combination of code written in `R`, python or other programming languages, scientific writing (including mathematical formula) and images to be included, and importantly, executed together to produce a document.[1]

1. In order to do this, they typically combine some form of text formatting or 'markup,' that ranges from very simple formatting conventions (for instance, the 'Markdown' format used in this book is much less complicated than HTML, and uses markup conventions readable as plain text and modelled on email (Gruber 2004);) to the highly technical (LaTeX, the de-facto scientific publishing format or 'document preparation system' (Lamport and LaTEX 1986) elements of which are also used here to convey mathematical expressions). They add to that blocks of code and inline code fragments that are executed as the text is formatted in order to produce results that are shown in ~~the text or inserted as figures in the text.~~

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 question can just about be pursued by scanning the matrix of plots, but not very stably since different data analysts might see different associations combining with each other there. Different statements or epistopics could be supported by the same figure.

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

Vector space expansion

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

etc., Stamey and other urologists more than a decade or so concluded that PSA is not a good biomarker for prostate cancer. Stamey writes in 2004:

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

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

Let us return to the equations for linear regression models (remembering that both C.S. Pierce and Andrew Ng advocate returning often to equations). The 'mainstay of statistics,' the linear regression model, usually appears diagrammatically in a more or less algebraic form:

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

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

Equations 3.1 and 3.2 express a plane (or hyperplane) in increasingly diagrammatic abstraction. The possibility of diagramming a high dimensional space derives largely from linear algebra. Reading Equation 3.1 from left to right, the expression $\hat{Y}$ already points to a set of calculated, predicted values, or a vector of y values, such as all the `lpsa` or PSA readings included in the `prostate` dataset. Similarly, the term X_j points to the table of all the other variables in the `prostate` dataset. Since there are 8 other

topology, or 'thought of the outside' (Deleuze 1988b), as a set of movements that sought to map the diagrams that generated a 'kind of reality, a new model of truth' (35). More recently, this topological thinking has been extended and developed by Celia Lury amongst others. In 'The Becoming Topological of Culture,' Lury, Luciana Parisi, and Tiziana Terranova suggest~~s~~ that 'a new rationality is emerging: the moving ratio of a topological culture' (Lury, Parisi, and Terranova 2012, 4)~~-~~. In this new rationality, practices of ordering, mode~~l~~ling, 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 mode~~l~~ling 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 epistopics and topologies together might, I suggest, help trace, map, and, importantly, diagram some of the movements into the data occurring today.

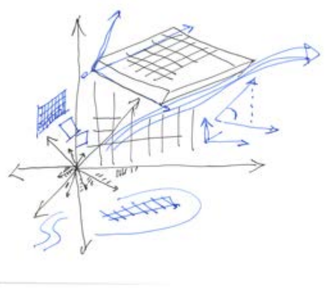

Figure 3.2: Vector space comprises transformations

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

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

In this expression, linear algebraic operations on the data shown as **X** calculate the coefficients $\hat{\beta}$ that orient a plane cutting through the vector space.[11] The derivation of the analytical 'ordinary least squares' solution

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

relies on some differential calculus as well as a range of linear algebra operations such as matrix transpose, inner product and matrix inversion, the details of which need not trouble us here. The relevant point is that equation 3.3 constructs a plane – a new vector – that traverses the density-shape of a dataset in its full dimensional vector space (nine dimensions in the case of `prostate`).[12]

Implicit vectorization in code and infrastructures

Vectorised transformations of data lie are the moving substrate of machine learning as it expands, but they are largely taken for granted as given space or commonsense ground. `R` coding practice instantiates vectorization in multiple ways, and is sometimes described as a 'vectorised programming language.' The vector space appears and operates just as directly in other programming languages designed for data practice (Octave, Matlab, Python's NumPy, or C++ Armadillo).

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

12. 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.

these differences in coding significantly affect human-machine relations. This simultaneity is only apparent, since somehow the underlying code has to deal with all the individual elements, but vectorised programming languages take advantage of hardware optimisations or carefully-crafted low-level linear algebra libraries.[14] More importantly, this is a different mode of movement. Operations now longer step through a series of coordinates that address data elements, but wield planes, diagonals, cross-sections and whole-space transformations. Vectorized code reduces both data and computational frictions. The real stake in vectorizing data is not speed but transformation. It makes working with data less like iteration through data structures (lists, indexes, arrays, fields, dictionaries, variables), and more like folding a pliable material. Such practical shifts in feeling for data are mundane yet crucial to the epistopic movements in data. [15]

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

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

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

15. A further level of vectorization appears in specific `R` constructs such as `apply`, `sapply`, `tapply`, `lapply`, and `mapply`. 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 a collection. ~~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

Vectorization also motivates increasingly parallel contemporary chip architectures, clusters of computers such as `hadoop` or `spark`, reallocation of computation to GPUs (Graphic Processing Unit), data-centre usage of FPGAs (Field Programmable Gate Arrays) and various other Cyclopean infrastructures of cloud computing. Many of these condensing and expanding movements of data are diagrammed in miniature in the `R` constructs as operators in vector space.

Lines traversing behind the light

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

Listing 3.2: Building a `prostate` model

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

Table 3.3 displays estimates of the coefficients or parameters $} that define the direction of a flat surface running through the vector space of the `prostate` data.[16] This new vector is a product of operations in the vector-

applying functions to functions seems strangely abstract.

16. From the epistopic viewpoint, the most obvious result of fitting a linear model is the production not of a line on a diagram or in a graphic. As we have seen, such lines cannot be easily rendered visible. Instead, the model generates a new column-vector

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

Table 3.3: Fitting a linear model to the extttprostate dataset

ized `prostate` data. Some vectorizing operations can be seen in `R` code in listing 3.2 (for instance, `as.matrix` or `scale(prostate_standard)`).

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

Listing 3.3: Closed form evaluation of linear model parameters

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

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`. The numbers of the table 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 vector space.

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

The vectorised table?

I started out from the observation that *Elements of Statistical Learning* mixes many datasets. The more abstract implications of vectorization and the forms of transformation movement it encourages and proliferates bring us back to the problem of how machine learning mixes datasets that span different settings. In short, vectorising computation makes the vector space, which we might understand as a resurgent form of the pre-Classical table,

17. This is an important differentiation: it is not typical machine learning practice to construct one model, characterised by a single set of statistics (F scores, R^2 scores, *t* values, etc.). In practice, most machine learning techniques construct many models, and the efficacy of some predictive techniques 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 epistopic 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 vector space. At the same time, the multiplicity of models multiplied by the machine learners becomes the topic of statistical analysis.

> ent worlds, non-consensual worlds, actively diverging worlds (Stengers 2005, 162)

If they cannot learn 'new habits,' what can functions learn? In some ways, Stengers would, on this reading, be taking a fairly conventional position on mathematical functions. They cannot learn or produce anything, only reproduce patterns implicit in their structure. Similar statements might be found in many philosophical writings on science and on mathematics in particular.[6] But throughout in her writing Stengers explicitly affirms *experimental practice*, much of which depends on functions and their operations (Stengers 2008). It might be better to say that she limits the agency of functions in isolation in order to highlight their specific power in science: 'celebrating the exceptional character of the experimental achievement very effectively limits the claims made in the name of science' (Stengers 2011, 376). (Limiting claims made for science might save it from being totally re-purposed as a techno-economic innovation system.)

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

> The reference of a mathematical function to an experimental

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 hybridi~~s~~es function and classification without any obvious contradiction.

We see something of this predictive desire from the basic mathematical expression for logistic regression in a situation where there are binary responses or $K = 2$:

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

(Hastie, Tibshirani, and Friedman 2009, 119)

Equation 4.2 encapsulates lines in curves. That is, the linear model (the model that fits a plane 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 relayed through the sigmoid function so that its output values no longer increase and decrease linearly. Instead they follow the curve of the logistic function, and range between a minimum of 0 and a maximum of 1, a range of values that map onto probabilities (as discussed in the next chapter 5. As usual, small typographic conventions diagram some of this transformation. 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 group or class (e.g. survival vs. death; male vs female; etc.).[9]

9. What does this wrapping of the linear model in the curve of the sigmoid logistic curve do in terms of finding a function? Note that the shape of this curve has no intrinsic connection or origin in the data. The curve no longer corresponds to growth or change in size, as it did in its nineteenth century biopolitical application to the growth of populations. Rather, the curvilinear encapsulation of the linear model allows the ~~left hand~~ side of the expression to move into a different register. The ~~left hand~~ side of the expression is now a probability function~~,~~ and defines the probability (Pr) that a given response value (G) belongs to one of the pre~~-~~defined classes ($k = 1, ..., K - 1$). 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 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

Curves and the variation in models

Whether or not the logistic function is a useful approximation to 'the function that underlies the predictive relationship between input and out' depends on how it relates input and output. 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 or 'response variable'. For a machine learner, finding a function means optimising function parameters on the basis of the data not deriving a formula. Machine learning is not a matter of mathematical analysis, but of algorithmic optimisation.[10]

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 presents

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. Even in machine learning, some function-finding through solving systems of equations occurs. For instance, the closed form or analytical 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 $\mathbf{1}$ along the diagonal, and $\mathbf{0}$ for all other values). All of these operations take place in the 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.

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. [12]

Observing costs, losses and objectives through optimisation

Faced with the impracticality of an analytical or mathematically closed form solution to the problem of finding a function, 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 varying solutions. A range of 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

12. 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.

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). It has a power-laden biopolitical history closely tied with knowledges and governing of populations in terms of morality, mortality, health, and wealth (see (Hacking 1975, 113-124). The key parameters here include μ, the mean and σ, the variance, a number that describes the dispersion of 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 statistically, functions such as the Gaussian distribution probabilise events as 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 population variations as random variables, that is, as probability distributions, was a significant historical achievement, one that continues to develop and ramify.[5] Random variables distribute probability in the world. When conceptualised as real 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,

Cauchy exponential, gamma, beta, hypergeometric, binomial, Poisson, chi-squared, Dirichlet, Boltzmann-Gibbs distributions, etc. (see (NIST 2012) for a gallery of distributions) — functionally express 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 northwest 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.)

5. 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 2015), random variables have become much more widespread in statistical practice due to changes in computational techniques.

```
    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)

The script draws out something of how the joint probability function in equation (5.3) probabilises a single word.[10] Not all machine learning models are so simple that they can be conveyed in 30 lines of code (including downloading the data and comments), but the script signals that nothing that occurring in probabilisation is intrinsically mysterious, elusive or indeed particularly abstract.[11] On the contrary, the power of classifiers operates through the accumulated counting, adding, multiplying (that is, repeated

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.

11. After fetching the dataset from a website, the code excerpted in 5.1 counts the number of emails in each category **spam** or **ham**, 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. In ~~P~~art 2, 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 ~~P~~art 3, 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. 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.)

Title	Year	Citations
A tutorial on Support Vector Machines for pattern recognition	1998	3510
Support vector machine classification and validation of cancer tissue samples using microarray expression data	2000	869
Choosing multiple parameters for support vector machines	2002	681
Classification of hyperspectral remote sensing images with support vector machines	2004	428
Comparing support vector machines with Gaussian kernels to radial basis function classifiers	1997	405
Support Vector Machines for 3D object recognition	1998	396
An assessment of support vector machines for land cover classification	2002	295
Drug design by machine learning: support vector machines for pharmaceutical data analysis	2001	273
A novel method of protein secondary structure prediction with high segment overlap measure: Support vector machine approach	2001	266
LIBSVM: A Library for Support Vector Machines	2011	246
The entire regularization path for the support vector machine	2004	223
A GA-based feature selection and parameters optimization for support vector machines	2006	223
The support vector machine under test	2003	219
Credit rating analysis with support vector machines and neural networks: a market comparative study	2004	218

Table 5.4: Most cited papers on support vector machines

The rapid rise to popularity of the support vector machine can be seen in the machine learning research literature, a very small slice of which appears in Table 5.4. 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 spans the normal biopolitical triangle of life, labour and language. The influence of the technique can also be seen in overlapping fields such as pattern recognition and data mining, where (Cortes and Vapnik 1995) and similar papers rank

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.

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 (`mnist`) are standard machine learning dataset. They have been frequently used to measure the performance of competing learning algorithms. In contrast to `iris`, the `mnist` is high dimensional. Each digit in the dataset is stored as a 16x16 pixel image. Image classification typically treats each pixel as a feature or variable in the input space. So each digit as represented by 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 `mnist` dataset has around 70,000. Aside from this dimensional growth, the handwritten digits aptly convey the blurring of differences. On the one hand, many people can easily recognise slight variations in handwritten digits with few errors. This is despite the many variations in handwriting that skew, morph and distort the ideal graphic forms of numbers.[27]

In their experiments with digit recognition (shown in figure 5.5, 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

27. Neural network researchers have heavily used the MNIST dataset. I discuss some of that work in chapter 7. 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 (Hastie, Tibshirani, and Friedman 2009, 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 digit~~al~~s 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 ancestral probabili~~s~~aton of such datasets.

not index new sources or kinds of data. Instead, the support vector machine transforms the vector space into a much higher dimension.

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). From the standpoint of pattern recognition, this often vastly augmented vector space should make it harder to locate patterns. A linear or planar decision surface in high dimensional space maps onto a curving even labyrinthine decision boundary when projected back onto the original vector space (see the curving decision boundaries in figure 5.4). 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 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.[29] Research in machine learning, whether it has been primarily statistical, mathematical or computational, countenances and addresses problems of non-linear classification through *dimensional expansion.*

The powerful augmentation characteristic of the support vector machine works through diagrammatic substitution. Consider the expression shown below in equation 5.7:

29. 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, labels, 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.

knowledge of the 3.8 billion years of evolution of species differences and population diversity. 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 association with knowledge economy. Prior to the whole genome sequencing projects initiated in the 1990s, biologists had never worked with genomes only with selected DNA sequences, especially those associated with genes and the proteins that they code. By contrast, the genome, with all its repeated, redundant, and slightly varying patterns of DNA, bears the traces of long evolutionary mixing and constitutes a hyper-complex functional process whose exquisite sensitivity to changing conditions – a slight change in light reaching a leaf cascades can be traced in patterns of DNA transcription – forms an extreme 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]

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.

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

As referentials, genomes pose a problem of unregulated abundance and seeming homogeneity. DNA sequences exist in great abundance (in databases, and increasingly, from the cheaper and more compact sequencing instruments), yet even determining how DNA sequence fragments should be ordered in a genome – let alone how they make sense as some biological function – is much harder. DNA sequences are assembled as genomes and genomic datasets via statistical models. '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 random sets of sequence fragments of various lengths that have to be assembled into a complete genome algorithmically. [4]

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.

4. 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) was not at the time understood as a machine learning problem. The task 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. 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.

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. They may just as well cross different thresholds of knowledge in governmental, market-focused, organisational or managerial operations.[7]

The plurality of applications can sometimes make it seem that machine learning arrives at the borders of different domains, and then proceeds to colonise local knowledges practices. The rule of materiality here would seem to be an epistemic *terra nullius* appropriation, in which existing knowledge forms are rapidly extinguished by machine learners. We have seen previously that ancestral communities of probabilisation orient the generalization of machine learning (see chapter 5). Research literature published on machine learning since the early 1990s clusters around problems of plethoric excess – image recognition, document classification, market behaviour (as in, working out what advertisement to show, or whether someone is likely to a buy a particular product, etc.). These problems position machine learning amidst regimes of communication, 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 genomics (arguably the successor of molecular

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.

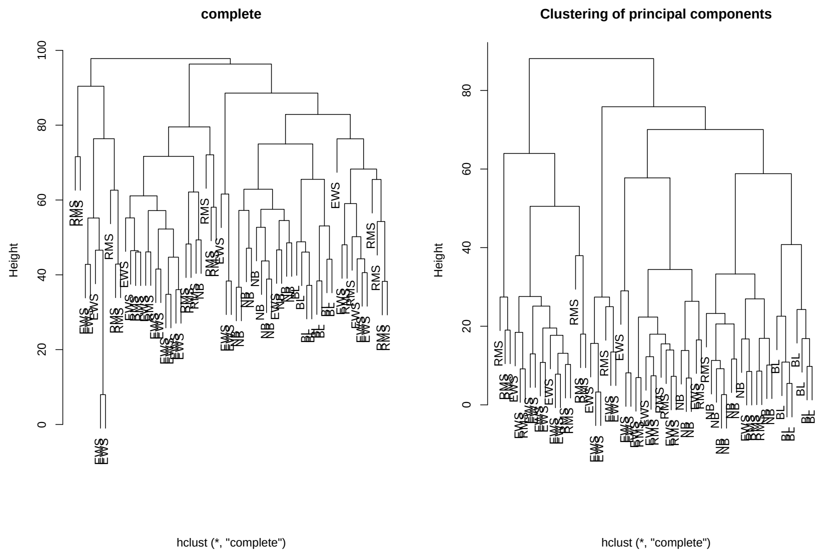

Figure 6.2: Hierarchical clustering of the SBRCT gene expression data

In contrast to the direct measurements of petals and sepals in the `iris` data, the `SRBCT` data incorporates and diagonally connects many levels of practice. The columns in table **??** refer to genes whose levels of expression in different samples are measured and then grouped by comparison to their levels in a reference sample (see the hierarchical clustering of the data

---

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.

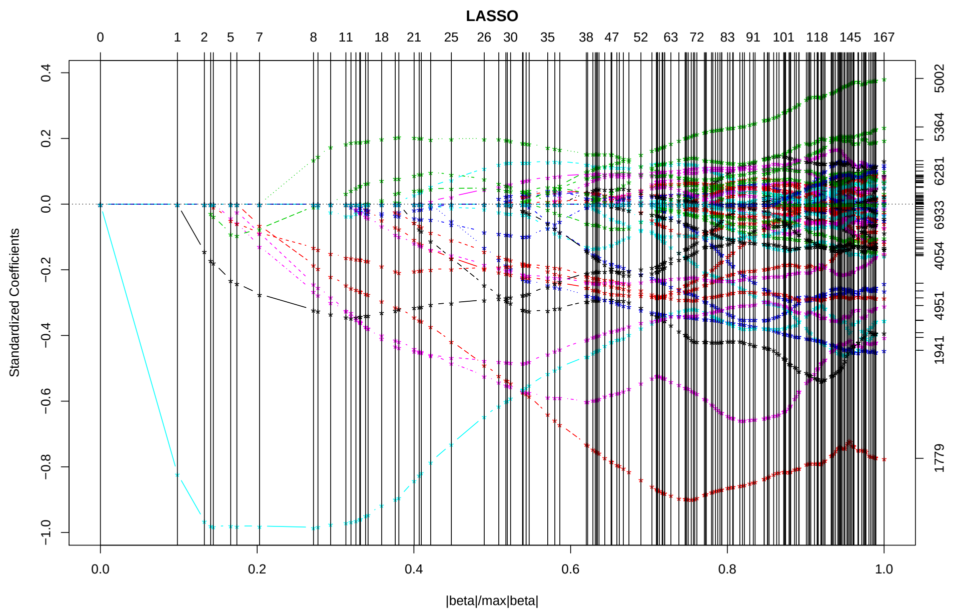

Figure 6.3: Shrinkage path of coefficients for Lasso regression on (Golub,1999) leukemia data

multinomial' model that uses only 384 genes, highlights a drastically reduced subset of the original object. A regularized genome of 384 genes suggest a much more targeted set of interventions than 16,000.

Proliferation of discoveries

Despite all the infrastructural cross-validation and regularization of plural expression, machine learning does not stabilise genomes as data objects. In many ways, it gives rise to further transformations and variations, and new sources of error.[11] If on the one hand, machine learners offer to regularize

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

transient multiplicities (such as gene expression in complex disorders), on the other hand, within genomics itself, machine learning exhibits considerable epistemic instability that itself needs to be regulated.

For instance, the US Food and Drug Administration has since 2003 conducted a study of data analysis techniques for microarray data:

> 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).

Phase I of the US Federal Drug Administration-led MAQC 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 its second phase known as MAQC-II starting in 2007, 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.

In the shift from MAQC-I to MAQC-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 transforms the 'feature space' (Parry

explore complex genetic traits and conditions. SNP-based DNA microarrays measure the occurrence of millions of SNPs in a given biological sample.

located at the intersection of network infrastructure, mathematics and human behaviour.[1] Mason, herself 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 - Machine Learning for Hackers* 2012), might personify such a 'wonderful person.' She is not a lone example. In mid-2016 Google announced a comprehensive program to re-train its software developers as machine learners (Levy 2016).[2]

'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. 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

1. In earlier work on machine learning (Mackenzie 2013b), 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). Subsequent~~ly~~ 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.

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.

decades, the development of the back-propagation algorithm as a way for a set of connected computational nodes to learn came with explicit infrastructural resonances.

The resonances between computational architectures and human cognition (centred on vision) became much more palpable from around 2006 when 'deep belief nets' appeared as a way of training many-layered neural nets implemented on much large computational platforms (Hinton, Osindero, and Teh 2006). These resonances continue to echo today and indeed attract much attention.[4] Like the advent of VSLI 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 6) and 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 have become just as important as loose or mutable infrastructural orders as epistemic instruments.

Oscillating between cognition and infrastructures, between people and machines, neural nets suggest a way of thinking not only about how 'long-term knowledge' takes shape today, but about subject positions associated with machine learning. As infrastructural reorganization takes place around

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 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 6. 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.

‘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’ (Hastie, Tibshirani, and Friedman 2009, 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 it automates what had previously been programmed by hand. Somewhat ironically, 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 unstable position of subjects in relation to neural nets are frequently discussed in contrasting terms by machine learners themselves.[6] They often

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 (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 1995; Bishop 2006). Among~~st~~ statisticians, Brian Ripley’s *Pattern Recognition and Neural Networks* (Ripley 1996), also highly cited, placed a great deal of emphasis on them. But these textbooks stand out 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.).

Competitions as examination

More or less directly, the observation of error rates converging towards minimum values assigns a subject position the role of controlling hyper-parameters such as γ, the learning rate. This seems a drastic curtailment of agency. The related feed-forward and back-propagation of errors focuses the machine learner subject, the 'wonderful people' of Hilary Mason's exhortation to developers, on error. At almost every step of its development as a field and in almost every aspect of its operation, competitions to observe and rank error rates bring human and machine learners together. In competitions, errors are not purely epistemic. They circulate within a wider economy of competitive optimisation that connect them to power, value and agency dynamics. 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 from such competitions about subject positions in machine learning?

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

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.

many-layered neural nets depends heavily on vectorised transformations of image data. Probably few other competitors in this competition would have had access to the Tesla K40 or 'NVIDIA GTX 980 Superclocked' GPU cards that 'Deep Sea' relied on.[13] 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' (Dieleman 2015). As is often the case, competition engenders populations of machine learners whose aggregate tendencies model optimum performance.[14] The 'DeepSea' team might epitomise machine learning subject positions. 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 in this case might have more to do with what other Kaggle competitors might be doing, as well as an awareness of cutting edge research leaders in image recognition techniques).

Ranked subject positions

'DeepSea' built models that classify images of more than a hundred kinds of plankton with few errors. In driving down error rates more than the hundreds of other competitors, they occupy a privileged subject position

13. 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.

14. 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.'

The predictive power of the model derives from the networked collective of elementary machine learners 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 layers – machine learners – to collectively learn from their rankings in tests of error.

As it disperses subject positions, the back-propagation of errors or optimisation also animates optimism about machine learning.[15] Machine learning hovers in potentiality 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 predictive, prospective and anticipatory. But this adjusting of weights carried out through the propagation of errors is also inherently a ranking or examination.

Human and machine learner differences can be re-drawn in two different directions. In one direction, machine learning operations assign a subject position focused on error rates. Vlad in his corner observing the neural nets occupied such a position. In the other direction, the subjects who operate the neural net in order to fit a model find themselves deeply caught up in a network of machine learners connected into parallel and layered architectures and operations. This feeding-forward, however, is regularized or narrowed down through examination and error, through back-propagation on various

15. The cultural theorist Lauren Berlant describes optimism as an 'operation':

> 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.