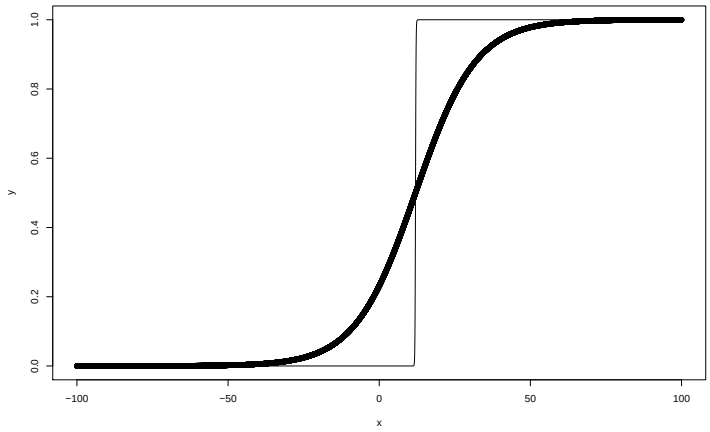


Figure 4.2: Logistic or sigmoid function for ~~2~~ different values of the parameter k

function, can be written as:

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

The logistic function (shown as ~~E~~quation 4.1 and as two curves in ~~F~~igure 4.2), as we will see, is very important in many classification and decision settings partly due to the *non-linear* shape that constrains vertical movement within the values (0 to 1), and partly because of the range of shapes opened up by the parameter K. How does a function such as the sigmoid function 'observe' anything? Here the curve itself and even the name 'sigmoid' is the best guide. ~~The S-shape of the sigmoid curve is a good guide to operations associated with curves.~~ The logistic function has quite a long history in statistics since that curve diagrams growth and change in various ways. ~~(~~As the historian of statistics J.S. Cramer writes: 'The logistic function was invented in the nineteenth century for the description of the growth of organisms and populations and for the course of autocatalytic chemical reactions' (Cramer 2004, 614).[8] 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.

8. The Belgian mathematician Pierre-François Verhulst designated the sigmoid function the 'logistic curve' in the 1830-40s (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.

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 describing and predicting growth in populations. Census data, clinical or laboratory measurements supplied the actual values of $f(x)$ at particular times, the x values. The task of the demographer, physiologist or economist was to calculate out the values of parameters such as k that controlled the shape of the curve.

Historically, then, the logistic function has a well-established biopolitical resonance. But note that the curves showing in Figure 4.2 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 4.2 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 curve when the scaling parameter k is larger transforms the function into a classifier, into a function that, as Breiman puts it, is equal to one of the numbers 0 or 1. In this setting, the function $f(x) = 1/(1+e^{-x})$ maps continuously varying numbers (the x values) onto a domain of 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 mapping between the x values sliding continuously and the binary difference pivots on the combination of the exponential function (e^{-x}), which

rapidly tends towards zero as x increases and rapidly tends towards ∞ as x decreases, and the dividend $1/1+\ldots$, which converts high value denominators to almost zero and low value denominators to one. This mapping between variations in x and 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 such as `cat` and `not-cat`. This realisation — that a continuously varying sigmoid function can map discrete outcomes — forms the basis of many machine learning classifiers.

The cost of curves in machine learning

I have been suggesting that experimentality in machine learning consists in coupling operational and observational functions. If operational functions move through or transform the data, observational functions render the effects of those transformations visible. How does this take place practically? The logistic function appears frequently in machine learning literature, prominently as part of perhaps the most vernacular machine learner, the logistic regression model (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)). In biomedical research, '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 superimpose the logistic function on

the linear model that we have been discussing already (see figure ?? and also chapters 2 and 3). As *Elements of Statistical Learning* puts it: '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). 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 $K = 2$:

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

(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 applications to the growth of populations. Rather,

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 'learn~~s~~' 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 itself there. For instance, in their analysis of the `South African coronary heart` disease data, Hastie and co-authors repeatedly model the risk of occurrence of heart disease using logistic regression. They first apply logistic regression fitted by 'maximum likelihood', and then by 'L1 regularized logistic regression' (Hastie, Tibshirani, and Friedman 2009, 126). The results of this function-finding work appear as tables of coefficients or ~~as~~ 'regularization plots'. As is often the case, *Elements of Statistical Learning* assumes that readers already understand conventional statistical usages of logistic regression. Discussion dwells instead on observing how values of the model parameter change as different variants of the model transform the data.

Figure 4.3 shows a series of lines. Plotted after the model transforms the data 366 times, each line sets out the changing importance

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 1 along the diagonal, and 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 instanc a dataset with a million rows and

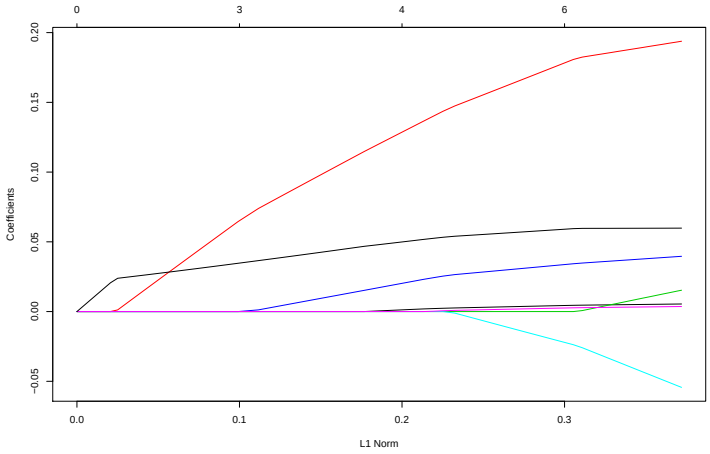


Figure 4.3: South African Heart disease regularization plot

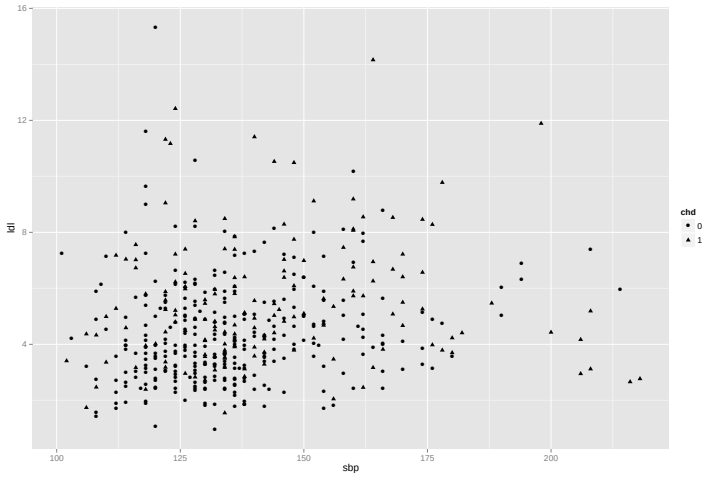


Figure 4.4: South African Heart disease decision plane

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 learning or function-finding diagrammed in figure 4.3 concerns variations in parameters and ways of automating the variation of parameters beyond that undertaken by modelling experts such as statisticians and scientists when they fit models to data.[11]

We saw that the classic statistical model of linear regression fits lines to the data through the linear algebra method of ordinary least squares (see chapter 3, equation 3.3). 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 more 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.[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

11. As we saw in chapter 3, the production of new tables of numbers that list the parameters of models actually transform the vectorised data into new subspaces (a line, plane, a surface). 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 slightly different transformations 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).

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

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 World War; 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 techniques of optimisation.

Optimisation is a practice of observation. '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).[13] In many machine learning techniques, the search for an approximation to the function that generated the data is optimised by reference to another function called the 'cost function' (also known as the 'objective function' or the 'loss function'; the terms are somewhat evocative of both economics and cybernetics). Machine learning problems are framed in terms of minimizing or maximising the cost function. 'Cost' or 'loss' takes the form 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).

The cost function compares predictions generated by a machine learner to known values in the data-set. Every cost function implies some measure of the difference or distance between the prediction and

13. 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.

the values actually measured. Cost functions in common use include squared error, hinge loss, log-likelihood and cross-entropy. 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 and hence machine learning in general occurs iteratively. Given a cost function, a machine learner can vary its parameters keeping in ~~view~~ or partially observing ~~whether~~ the cost function increases or decreases. Just as the logistic function wraps the linear regression model in a sigmoid curve that switches smoothly between binary values, the cost functions diagram model parameters (usually noted as β) in relation to known responses or output values in the data. If there is learning here, it ~~does derive~~ from mathematic forms or higher abstraction. Cost functions diagram relations between models, and render their predictive reference through the negative feedback loops described by Norbert Wiener (Wiener 1961). Importantly, these feedback loops are not closed mechanisms but places from which variations can be viewed.

For instance, the log-likelihood function, a typical and ~~widely-used~~ cost function associated with logistic regression is defined as:

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

where

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

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. ~~They substitute in~~ through the subsidiary function $h_\beta(x)$, the logistic function function encapsulating a linear function $\beta^T X$~~.~~ Second, the function defines a goal of maximising the overall value of the expression~~,~~ $J(\beta)$ as a function of variations in the parameters β. ~~The *min* describes the results of the repeated application of the function.~~ Third, the heart of the cost function is balancing ~~of~~ two tendencies: it adds ($\sum$) all the values where the probability of the predicted class of a particular case $h(x_i)$ matches the actual class y_i~~,~~ 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 through optimisation~~,~~ 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

We have some sense of how a function can be configured as an observer~~,~~ but little sense of how they manage variations. Many optimization techniques rely on differential calculus and particularly the calculus of variations to maximise or minimise the value of a cost function. In fact, loss functions are often chosen on the basis of their differentiability. ~~One widely used~~ optimisation algorithm called ‘gradient descent’ ~~is quite easy to grasp intuitively~~. In neural nets and deep learning, gradient descent (or ascent) occurs on an increasingly vast scale. ~~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:

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

The version of the algorithm shown in algorithm 4.4 is called 'stochastic gradient descent'. Archaeologically, in presenting such formula, the point is not to read and understand them directly but to characterise the enunciative function that regulates them. Practical understanding 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 2. 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 via functions supplied by software libraries (e.g. `GradientDescentOptimizer` in the `TensorFlow` library or `StochasticGradient` in `torch`).

Given that equation 4.4 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 ((*Lecture 3 | Machine Learning (Stanford)* 2008); see figure 4.5), mixes an algorithmic set of operations with function

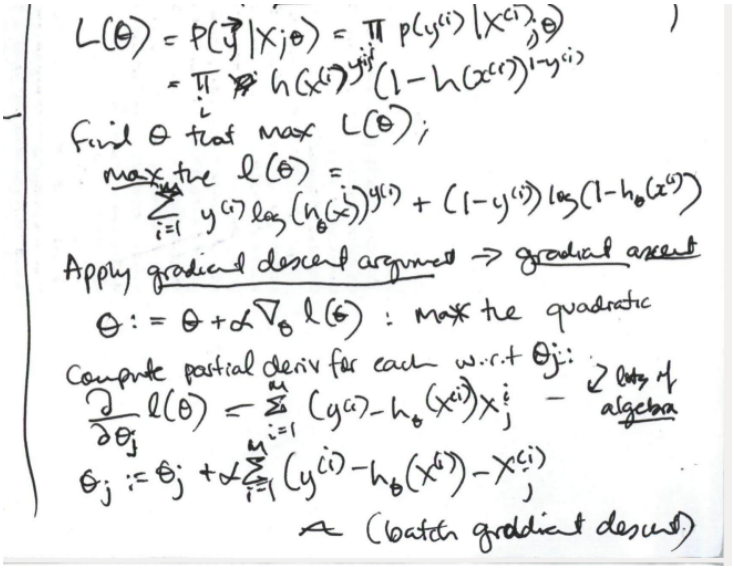


Figure 4.5: Gradient ascent for logistic regression

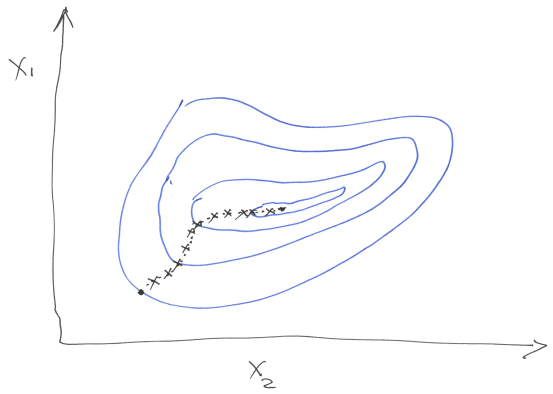


Figure 4.6: Stochastic gradient descent path

notation. We see this in several respects: the formulation includes the imperative 'repeat until convergence'; it also uses the so-called 'assignment operator' := rather than the equality operator =. The latter specifies that two values or expressions are equal, whereas the former specifies that the values on the ~~right hand~~ side of the expression should be assigned to the left. Both algorithmic ~~forms — repeat~~ until convergence~~,~~ and assign/update ~~values — owe~~ more to techniques of computation than to mathematical abstraction.

In gradient descent, we see functions acting as partial observers.

The specification for the gradient descent algorithm brings us to the scene where ongoing transformation of data from irregular volume to plane can be observed. 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 the rate at which one variable changes in relation to another; that is, as the rate at which the cost function $J(\beta)$ changes with respect to the different values of β.[14] Much learning in machine learning pivots on the observation of rates of change of a cost function in relation to its 'arguments,' the j-dimensional vector space defined by β, the model parameters. The partial derivatives in the gradient descent algorithm observe the direction in which the value of the cost function reduces. Each iteration of the algorithm reduces or increases the parameters β of the model in the direction of reduced cost, and perhaps less error. Importantly, the derivative of a sigmoid (or logistic function) $\sigma(x)$ is given by $\sigma(x)(1-\sigma(x))$, which means that the partial derivatives of a cost function will be easy to compute.

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

The power to learn

The power of machine learning to learn, its power to epistemologize, pivots around functions in disparate yet connected ways: the transformation of data through operational functions that map new sub-spaces in vector space and in the observational functions that algorithmically superimpose new constraints — ~~cost~~, loss or objective functions — ~~that~~ direct an iterative process of optimisation. Machine learning diagrammatically distributes learning in the operational human-machine formation. People look at curves for evidence of convergence, functions compress data into functions that support classification or predictions, and algorithms observe gradients or rates

of error in relation to model parameters. 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 cost functions, as they seek to minimize differences between the predicted values and the known values found in the vector space, control variations in the model. 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 transform data in the 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. ~~Universality~~ (*apropos* master algorithms), whether fully supervised or completely unsupervised, is impossible here.

Amid~~st~~ this endemic partiality, we can begin to understand the multiplication of machine learners and the mirage of universality. Machine learners are functions that transform data and observe the effects of those transformation in learning to classify, predict, and rank. But the function that defines a '~~'~~machine learner~~'~~ contracts a range of partial observers relaying values and changes to each other. The operational power of machine learning depends on the diagrammatic and sometimes experimental relays 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, the enunciative function that connects different elements in the production of consequential classifications and predictions, generating operational statements in

fields of knowledge.

We are in a slightly better position to understand now how there can be many machine learners but a relative sparsity in the production of statements. Gradients — continuous variations in rate — are useful because they generate many functions, many approximations within one operational process, within one enunciative function. The profusion of machine learners, the 'bewildering variety' that Domingos and others celebrate and flag up, can be seen as the effect of an operational formation predicated on approximation through variation.

In his account of Foucault's 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 1988b, 35)

Functions in machine learning are 'intersocial' in the sense that they bring together very different mathematical, algorithmic, operational and observational processes. The sigmoid function switches the geometry of the linear model over into the calculation of probabilities and classification, but also figures heavily in the computability of partial derivatives. 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 relays that connect 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.

The machine learning diagram, like any functioning, harbours the potential for invention. Describing the application of machine learning to biomedical and clinical research, James Malley, Karen Malley and Sinisa Pajevic contrast it to more conventional statistical knowledges:

> 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)

Novel structures and functions in 'our data' are precisely the functions that machine learning technique seek to learn. Could new habits or actively diverging worlds that Stengers calls for appear amidst this quasi-iterative pursuit of optimisation and convergence? This is a terrain for critical thought to explore. A function in isolation never learns. But when watched or observed, even virtually, divergence has some chance. To the extent that machine learners relay references experimentally between things and people, mobilising the production of statements and visibilities across different elements, divergence remain possible.

Year	Title	Keywords	Citations
1995	Computed Tomography Imaging Spectrometer Experimental Calibration And Reconstruction Results	imaging spectrometry; computed tomography; experimental point-spread-function characterization; central-slice theorem; missing cones	80
1997	Comparing Support Vector Machines With Gaussian Kernels To Radial Basis Function Classifiers	clustering; pattern recognition; prototypes; radial basis function networks; support vector machines	353
1997	The United Adaptive Learning Algorithm For The Link Weights And Shape Parameter In Rbfn For Pattern Recognition	neural networks; recognition; radial basis function network; gaussian kernel function; shape parameter; forgotten factor; recursive least squares; adaptive gradient descending; one-dimensional image	30
1999	Learning By Discovering Concept Hierarchies	function decomposition; machine learning; concept hierarchies; concept discovery; constructive induction; generalization	23
2000	Block Coordinate Relaxation Methods For Nonparametric Wavelet Denoising	basis pursuit; block coordinate relaxation; function estimation; interior-point; optimization	90
2003	Support Vector Machine With Adaptive Parameters In Financial Time Series Forecasting	back-propagation (bp) neural network; nonstationarity; regularized radial basis function (rbf) neural network; support vector machine (svm)	181
2003	A Knowledge Based Approach To Response Surface Modelling In Multifidelity Optimization	multifidelity modelling; knowledge-based neural networks; kriging; expensive function optimization	31
2005	Reliability And Validity In Comparative Studies Of Software Prediction Models	software metrics; cost estimation; cross-validation; empirical methods; arbitrary function approximators; machine learning; estimation by analogy; regression analysis; simulation; reliability; validity; accuracy indicators	51
2006	Dictionary Based Stochastic Expectation Maximization For Sar Amplitude Probability Density Function Estimation	finite mixture models (fmms); parametric estimation; probability density function (pdf) estimation; stochastic expectation maximization (sem); synthetic aperture radar (sar) images	31
2006	Soft Sensor Development For Fed Batch Bioreactors Using Support Vector Regression	artificial neural networks; bioreactor; soft-sensors; support vector regression; multilayer perceptron; radial basis function network	41
2006	A Wavelet Network Model For Short Term Traffic Volume Forecasting	wavelet networks; back-propagation neural networks; radial basis function neural networks; levenberg-marquardt algorithm; traffic volume forecasting	30
2006	Improving Rbf Networks Performance In Regression Tasks By Means Of A Supervised Fuzzy Clustering	fuzzy clustering; fuzzy c-means; radial basis function neural networks; linear regression models	32
2007	Choosing Parameters Of Kernel Subspace Lda For Recognition Of Face Images Under Pose And Illumination Variations	gaussian radial basis function (rbf) kernel; generalization capability; kernel fisher discriminant (kfd); kernel parameter; model selection	33
2007	Localized Generalization Error Model And Its Application To Architecture Selection For Radial Basis Function Neural Network	localized generalization error; network architecture selection; radial basis function neural network (rbfnn); sensitivity measure	51
2007	An Efficient Strategy For Extensive Integration Of Diverse Biological Data For Protein Function		23

Year	Title	Keywords	Citations

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

5

$N = \forall X$ Probabilisation and the Taming of Machines

In the final pages of *The Taming of Chance*, Ian Hacking describes the work of the philosopher Charles Sanders Peirce in terms of a twin affirmation of chance. First, Peirce, following the work of the psychophysicist Gustav Fechner and before him the astronomer-sociologist Adolphe Quetelet, re-enacts the normal curve.[1] The 'personal equation,' the variation in measurements made by any observer, becomes 'a reality underneath the phenomena of consciousness' (Hacking 1990, 205). Peirce's belief in absolute chance or a stochastic ontology, 'a universe of chance' as Hacking puts it, continued a series of 'realizations' of curves, in which astronomical, social, biological and finally psychological variations were all understood as generated by processes of chance. Second, 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). In the century or so since, what happened to the thorough-going affirmation of statistical thought and probabilistic practice epitomised 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

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

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 the opening lines of the preface to the First Edition of *Elements of Statistical Learning*, Hastie, Tibshirani and Friedman describe the altered situation of statistics:

> 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 the door of statistics in recent years has not only been more data but also machine learners. 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 *probabilised* machine learners, or injected a substratum of chance that flows directly from their operation. To grasp this we need to determine what role randomness, change and the probabilistic distribution of elements and events 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 taming of data — and machines — in statistical practices of machine learning today.

Data reduces uncertainty?

The broadest claim associated with machine learning hinges on the simple expression shown:

$$N = \forall \boldsymbol{X} \tag{5.1}$$

In Equation 5.1, N refers to the number of observations (and hence the size of the dataset), the logical operator $\forall$ means 'all' since this is the level of inclusion which many fields of knowledge in science, government, media, commerce and industry envisage, and $\boldsymbol{X}$ refers to the data itself arrayed in vector space. Note that this expression leaves some things out. Y, the response variable, for instance, may or may not be known or part of the data $\boldsymbol{X}$. While both the expansion of data in the vector space and the machine learners that transform and observe it have appeared in previous chapters, I focus here on changes in probability practices associated with machine learning, and in particular, $N = \forall \boldsymbol{X}$, the claim that with all the data, the production of knowledge fundamentally changes.

The claim that with $N = \forall \boldsymbol{X}$ the nature of knowledge 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 settings in the course of the vignettes and teeming comparisons that 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

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 — epistopic construction of 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.

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. Machine learning remains almost completely invisible as a practice of transforming data in the name of knowledge. ~~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 data practices 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).

Without a sense of how statistical practices animate and configure key features of 'crunching the data' to make predictions, it becomes hard to see how the 'revolution' takes place. Just as ~~nineteenth century~~ statistics transformed many measurements into population attributes (~~for instance~~, mean as the ideal or abstract property of a population), the shift between n and $\forall \boldsymbol{X}$, between some and all, a shift ~~very much~~ dependent on machine learning, internalizes, I will suggest, population attributes into the operations of machine learners. 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 'probabilisation,' a pleonasm that refers ~~that facet~~ of the operational formation that ~~renders knowledge~~ in terms of probabilities.

Machine learning as statistics inside out

The argument mimics Hacking's. In *The Taming of Chance*, Hacking argues that modern statistical thought transposed a way of calculating errors in experimental measurements and astronomical observations into the real and ~~constitute~~ attributes of populations understood as processes of reproductive growth. This transposition or inversion relied on four intermediate steps passing through the development of a probability calculus (particularly the work of Jacob Bernoulli and the binomial or heads-tails probability distribution in the 1690s (143)), the accumulation of large numbers of measurements (the most famous being the chest measurements of soldiers in Scottish regiments, but these were only one flurry amid~~st~~ an avalanche of numbers in the 1830-1840s), the emergence of the 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, among~~st~~ others, astronomers (Hacking 1990, 111-112). As Hacking observes, coins, suicides, crime, chest measurements, and astronomical observations all pile up in a statistical aggregate ~~which~~ remains, although somewhat altered, indelible in contemporary statistical knowledges, particularly in its frequent recourse to notions of population, probability and distribution. 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 a population.

Machine learning reverse-engineers the invention of modern statistical thinking. It takes back the 'real quantities' — probabilities ~~that~~ modern statistics had attributed to the populations in the world and distributes them to devices, to 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. Although it relies heavily on probability distributions, machine learning is a fat-tailed distribution of probability.

The swapping or re-distribution is not a simple mirror-image reversal, as if machine learners mistake devices for a population. Machine learning constantly takes statistical thinking as a basic condition for its operations and devices. When *Elements of Statistical Learning* states that (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 operational practices in machine learning as a field of knowledge-practice show the marks of increasingly strong commitment to the reality of the statistical, and to the ongoing probabilisation of machine learners.

What is probabilisation in practice? Reading and working with machine learning techniques usually means encountering and responding to apparatus drawn from statistics, but the apparatus is 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 a more advanced guide such as *All of Statistics* (Wasserman 2003), where statistical tests (t-test, chi-squared test, etc.), hypothesis testing, and analysis of uncertainties (confidence

parametric	non-parametric
bias	variance
prediction	inference
generative	discriminative

Table 5.1: Some structuring differences in machine learning

intervals, etc.) order the exposition, machine learning textbooks rely on a conceptual apparatus curiously stripped of statistical tests and measurements. Statistical underpinnings may be fundamental, but this does not mean that machine learners simply automate statistics.

Instead, a basic set of contrasts or indeed oppositions that owe much to probabilistic thinking order, compose, associate and link the statements of machine learners. 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 models whereas logistic regression or support vector machines are *discriminative*); parametric and non-parametric 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 $N = \forall \boldsymbol{X}$. Because they anchor basic operations of machine learning in probability, 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.

Distributed probabilities

While these structuring differences deeply shape practice in machine learning, the underlying operator that allows swapping between knowledge and the world, between events and devices, is probability, and in particular, functions that describe variations in populations, probability distributions. Probability distributions both map population variations and, as we will see, multiply the number of things that count as populations.

The normal distribution pervades nineteenth century statistical thinking as it affects populations across law, medicine, agriculture, finance and not least, sociology as a domain of knowledge. Normal distributions appear in countless variations in scientific, government and institutional settings as functions that map events, measurements, observations and records to evidential probability quantities.[3]

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

The function shown in equation (5.2) expresses the probability of a given value of the variable x given a population whose variations (with respect to x) can be expressed in terms of two parameters, μ and σ, the mean and variance. This is the so-called normal or Gaussian distribution.[4] 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). 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

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

4. Dozens of differently shaped probability distributions map continuous and discrete variations to real numbers. Other probability distributions — normal (Gaussian), uniform, 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

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, positioning, partitioning, sectioning, serialising or queuing operations has received much more attention in critical thought (particularly in the many uses of Foucault's concept of disciplinary power (Foucault 1977)), but in almost every setting, distribution in the sense of counting, apportioning and weighting of different outcomes also operates. This constant interweaving of spatial, architectural, logistical and functional processes has energised statistical thought for several centuries.[6] 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

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.

6. '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).

~~of referring to a~~ property of populations, the normal curve has not only described but modulated and re-shaped populations (in terms of health, morality and wealth).

~~If~~ functions such as equation (5.2) have persisted for so long as elements of population governmentality or biopolitics, what happen to them in machine learning? The pages of a book such as *Elements of Statistical Learning* show many signs of an ongoing invocation of probability distributions. We could simply observe their abundance. Hastie and co-authors invoke probability distributions. They speak of ~~'Gaussian mixtures,' 'bivariate Gaussian distributions,' standard Gaussian,''Gaussian kernels,' '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.) Events, things, properties, operations, functions, and attributes all associate with probability distributions.

The multiple invocations of probability distributions ~~attests to the~~ variety of events (occurrence of cancer, occurrence of the word 'Viagra' in an email, a click on a hyperlink, etc.) ~~map~~ to real numbers. Despite the sometimes dense mathematical diagrammaticism, 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.[7] Whatever inferences and predictions become possible, probability distributions are a crucial control surface for machine learning understood as a form of movement through data. In contrast to the endowment of living aggregates such as populations with probability that we see in the biopolitical history of statistics (and later in natural sciences such as physics and biology), statistical machine learning increasingly constitutes devices as populations via

7. Machine learners adjust these parameters in different ways. For instance, parametric and non-parametric models (see table 5.1) 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.

probability distributions.

Naive Bayes and the distribution of probabilities

How could machine learners become a population? The mathematical expression for one of the most popular of all machine learning classifiers, the Naive Bayes classifier, stands out for its probabilistic simplicity and seeming lack of 'moving parts':

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

(Hastie, Tibshirani, and Friedman 2009, 211)

Some machine learners are so simple that they can be implemented in a few lines of code. Along with the perceptron, linear regression, and k nearest neighbours, the function shown in equation (5.3) is one of the simplest ~~one~~ to be found in most machine textbooks yet easily adapts for ~~high dimensional~~ data, the kind of data associated with contemporary network infrastructures, scientific instruments, online communications and $N = \forall \boldsymbol{X}$ in general.[8] ~~Even though~~ 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 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 an estimate of the probability that a given message, image or event is an instance of class j, p different features are taken into account. (The subscript k indexes the p dimensions of the vector space.) The subscripts $k = 1$ on

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' (463). Like the Naive Bayes classifier, the equation for k nearest neighbours is simple:

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

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 vector space.

the $\prod$ operator, and k on the data X_k indicate that the Naive Bayes classifier 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. Put in the language of probability calculus, the classifier produces a probability density $f_j(X)$ 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).

The Naive Bayes classifier directly invokes 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 to the application 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 a class to the occurrence of combinations of variables or features. Naive Bayes links variables 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 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 architecture based on the concepts of conditional probability and joint probability; it calculates a

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.

probability density function $f_j(X)$ or probability distribution for each possible class of things as a combination of the probabilities of all the many features or attributes of populations that come together in data. It makes a drastically naïve assumption that features or variables are statistically independent of each other, 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 such as independence do not necessarily weaken the referential grasp of machine learners on the world, but in certain ways allow them to reconfigure the operations of machine learners as a population of learners.

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

$\forall N$ can be a bother. In *Doing Data Science*, Rachel Schutt and Cathy O'Neill furnish a `bash` script (that is, command line instructions) to download the well-known `Enron` email dataset and build a Naive Bayes classifier that labels email as spam or not. In many ways, this is canonical machine learner pedagogy. For Naive Bayes, email spam detection has become the standard example (Andrew Ng uses it in CSS229, Lecture 5 (*Lecture 1 | Machine Learning (Stanford)* 2008)). In this setting, machine learners operate as filters coping with too much communication.

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.. 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:

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

Such communications, with their mixture of solicitation and imperative are familiar to anyone who uses email. How does Naive Bayes probabilise their differences? How do they become X or even $f_j(X)$ in the Naive Bayes classifier? The code that *Doing Data Science* supplies is instructive:

Listing 5.1: A Naive Bayes classifiers for `enron` email

```
#!/bin/bash
# description: trains a simple one-word naive bayes spam
# filter using enron email data
# usage: ./enron_naive_bayes.sh <word>
# author: jake hofman (gmail: jhofman)

### PART 1
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 2
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 3
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)

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 adding) and dividing (that is, multiplying by parts or fractions) constrained by the joint probability distribution. Probability re-distributes things such as emails or documents as, in this case, events in a population of words. The

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

Naive Bayes classifies endows every word in the `Enron` dataset 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`, 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 chance of being spam, ~~while~~ 'sexy' has a chance of 1. 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

~~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. Like most machine learning techniques encountering complex realities, classifiers such as Naive Bayes ignore many obvious structural or semiotic features of emails as documents (~~for instance~~, word order~~,~~ or co-occurrences of words). Yet this ~~very~~ artificiality or limitation in their reference to the world allows machine learners to appear in many different guises. Despite their simple architecture, Naive Bayes classifiers have been surprisingly successful. Many machine learners transform vectorised data into probability distributions populated by fields of random

variables in process of change. They render all things as populations.

	Year	Title
272	2000	Naive Bayes for regression
268	2002	On Discriminative vs. Generative classifiers: A comparison of logistic regression and naive Bayes
927	2004	Molecular similarity searching using atom environments, information-based feature selection, and a naive Bayesian classifier
264	2004	Some theory for Fisher's linear discriminant function, 'naive Bayes', and some alternatives when there are many more variables than observations
269	2004	Augmenting naive Bayes classifiers with statistical language models
1010	2004	Enrichment of extremely noisy high-throughput screening data using a naive Bayes classifier
271	2004	Combination of a naive Bayes classifier with consensus scoring improves enrichment of high-throughput docking results
1009	2005	Not so naive Bayes: Aggregating one-dependence estimators
265	2006	Prediction of protein homo-oligomer types by pseudo amino acid composition: Approached with an improved feature extraction and Naive Bayes Feature Fusion
872	2006	Combining multi-species genomic data for microRNA identification using a Naive Bayes classifier
1023	2006	Enrichment of high-throughput screening data with increasing levels of noise using support vector machines, recursive partitioning, and Laplacian-modified naive Bayesian classifiers
711	2008	Ligand-Target Prediction Using Winnow and Naive Bayesian Algorithms and the Implications of Overall Performance Statistics
840	2009	Feature selection for text classification with Naive Bayes

Table 5.2: Most cited Naive Bayes publications 1945–2015

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 5.2 shows 30 of the most cited Naive Bayes-related scientific publications.[12] The list of titles sketches a double movement. On the one hand, we see the typical diagonal forms of accumulation or positivity 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 during these years, an intensified probabilisation of machine learners that strongly affects their ongoing development (leading, for instance, to the much more document-oriented, heavily probabilistic topic models appearing in the following decade (Blei, Ng, and Jordan 2003)).

12. 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.

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

> 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).

Similar formulations can be found in most of the machine learning books and instructional materials. This appropriateness relates directly to $\forall \boldsymbol{X}$, and the expansion of the vector space. 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 balloon hundreds of thousands because every unique word adds a new dimension to the 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 the increasing availability of data to train machine learners on. I want to explore two other contrasts as ways of viewing the probabilising processes at work in Naive Bayes. 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 sources of error.

Ancestral probabilities in documents: inference and prediction

Why is 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 mentioning 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 regularity in production of statements? Admittedly spam, and spam trying to sell Viagra in particular, has been a ~~very~~ familiar part of most email 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 mundanely shared. 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~~ other would be scientific publications. Many more recent machine learners train as classifiers on scientific publications (Blei and Lafferty 2007)~~)~~.

From an archaeological standpoint, the reiteration of email spam filtering using Naive Bayes is the effect of another process~~, a process~~ akin to the attribution of probability distributions to populations in the ~~nineteenth~~ century. Like many machine learners, Naive Bayes has one important lineage derived from the problem of classifying and retrieving documents amid~~st~~ archives. The operational practice of document classification is specified in the element of the archive. 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,

> 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 reaching back to the mid-20th century.[13]

Early attempts to use what is now called Naive Bayes in the early 1960s re-iterated 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:

> 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 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

13. 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-).

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 document and 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 simplifying assumptions about language and meaning do not vitiate the Naive Bayes classifier. Whenever machine learners announce the unlikely efficacy of classifiers, we might attend to the ways in which previous 'ancestral probabilisations' and archival constitution of the domain in question prepare the ground for that success.

Statistical decompositions: bias, variance and observed 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

artificiality of Naive Bayes. The classifiers generates highly arbitrary probabilities of document class membership, yet these arbitrary probabilities still 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 heavily in 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 decomposition' (Hastie, Tibshirani, and Friedman 2009, 223).[14] 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

14. Another source of error, the 'irreducible error' (37) is noise that no model can eliminate.

function and the actual process that generated the data. Note that both sources of error in the bias-variance decomposition derive from transformations of the data. Variance affects how the model encounters the world (as a set of small samples or as, at the other end, a massive $N = \forall \boldsymbol{X}$ dataset). Bias relating to how the model 'apprehends' the data (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.).

Even with all the data, machine learning cannot fully circumvent the tensions between the different errors at work in the bias-variance decomposition. Yet, sources of error do not always prove harmful. The success of Naive Bayes (and k nearest neighbours classifier) runs counter to 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:

> Certain types of (very high) bias can be canceled by low variance to produce accurate classification (Friedman 1997, 55)

A rather elaborate set of concepts and techniques address the bias-variance decomposition in the context of data availability. 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. Machine learners in such settings still encounter the bias-variance trade-off as they select some data for training and some ~~data~~ for testing. This trade-off has to deal with the fact that training errors — the observed difference between what the model predicts and what the training data actually shows — are not a good guide to 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 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 (e.g. cross-validation (Stone 1974), bootstrap (Efron 1979), expectation-maximization (Dempster, Laird, and Rubin 1977)).

A daunting field of concepts, themes, techniques and methods all gravitate to the threshold of probabilisation. They invoke in some cases sophisticated mathematical or statistical constructs. They also ~~very~~ often rely on computational iteration or infrastructural scale to optimise parameters in models whose underlying intuitions remain quite straightforward (as in a linear regression or Naive Bayes). In some cases, the implementation of a model may be ~~very~~ simple, but analysis of how the machine learner manages to curtail a source

of error such as bias or variance entails much more sophisticated statistical understanding. Many analyses of how a model becomes a 'useful approximation' reconfigure ~~treat~~ the models ~~themselves~~ as members of a population whose variations and uncertainties, whose tendencies and predispositions must be sampled, tested and monitored. The bias-variance decomposition points to an irreducible friction in the way that machine learning structures differences in the world.

Does machine learning construct a new statistical reality?

Following a broadly Foucaultean line of argument, Hacking proposes that statistical thinking and practice in the ~~nineteenth~~ and early ~~twentieth~~ century ontologically 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 — distributes probability throughout an operational formation? I have suggested that an ancestral probabilisation of domains and the statistical decomposition of error come together in statistical machine learning. The bias-variance decomposition includes both tightly bound points and certainly relatively free or unbound points, as we saw in the case of the Naive Bayes classifier in its encounter with data. It generates highly erroneous probability estimates but performs well as a classifier.

Viewed diagrammatically, unbound points matter greatly to the relations of force at work in a knowledge-power conjunction. Probabilisation gives machine learning a relation to its own plurality, to the tendencies of its models to proliferate and vary. Every attempt to construct a machine learner in a given setting draws on both the

re-iteration of ancestral probabilities (that is, prior structuring of settings in conformity with some probability distribution) or on the many interactive adjustments, re-distributions and re-samplings of the data *and* transformations of the models associated with the bias-variance decomposition.

Mayer-Schönberger and Cukier argue that having much data or all data ($N = \forall \boldsymbol{X}$) re-bases knowledge. 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 archive or 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 wider inferences (11). Their discounting of statistical sampling as a concept 'developed to solve a particular problem at a particular moment in time under specific technological constraints' (Mayer-Schönberger and Cukier 2013, 31) does not, however, accommodate the operational practices of sampling that pervade machine learning, particularly in the forms of probabilisation.

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.

Whether or not someone uses Naive Bayes, a topic model, neural networks or logistic regression, does not greatly alter the processes of probabilisation. Random variables, probability distributions, errors and model selection practices crowd in around and re-configure machine learners as members of a population generating statements. In many ways, the Mayer-Schönberger and Cukier account bobs in the wake of the enterprise-wide accumulations of data. They pay so much attention to the capital potentials of data accumulation that they cannot easily attend to the question of how machine learners

probabilise ~~that~~ data. Sampling, estimation, likelihoods, and a whole gamut of dynamic relationships between random variables in joint probability distributions reassert themselves amid~~st~~ a population of models. The data may not be sampled, but models moving through the high-dimensional vector spaces opened up by having ~~'all'~~ the data transform it probabilistically. ~~While~~ not all machine learners are strictly speaking probabilistic models,[16] machine learners relate to themselves and the data as populations defined by probability distributions.

Machine learning inhabits a reality that had already introjected statistical realities at least a century earlier, whether through the social physics of Quetelet, the biopolitical norms of Francis Galton and his regression to the mean (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. Assembling an aggregate reality of many devices, machine learning inverts probability distributions. In this inversion, probability distributions, which had become the operational statement and model of truth for many different kinds of populations, fold back or re-distribute themselves into devices such as machine learners whose variations and uncertainties become populations. 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 population of models of populations.

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.

6

Patterns and difference

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

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

Do machine learners generate new patterns of difference? Should we hold machine learners accountable for their claims to recognise patterns in data in the same way we hold experimental scientists accountable for their factual claims?[1] This chapter explores two major machine learning treatments of pattern dating from the last decades of the twentieth century from the standpoint of differences. I suggest that what counts as pattern changes in machine learning over time. While much machine learning strains to identify differences in terms of differences of degree, the practice of pattern-finding itself harbours differences of kind. For critical thought, the connection between pattern and differences is particularly important, since if machine learning changes what counts as pattern, this will also affect the recognition or articulation of differences. We have seen the emergence of the vector space and its vectorised transformations,

1. Many authors have suggested that algorithms should be the focus of more attention. The sociologist 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).

the multiplication of operational functions and their associated partial observers, and then the probabilisation that distributes machine learners into populations of error-sensitive learners. What in this diagram and in the forest-like growth of techniques, projects, applications and proponents, allows us to make sense of what happens to differences in machine learning?

Across vectors, functions and populations, the diagram of machine learning weaves and knots many points of emergence, continuity and conjunction. I view the formidable accumulations of infrastructure, devices and expertise accrediting around machine learning as multi-faceted abstractions, where abstraction is understood diagrammatically as a concretising entanglement of references. Three highly developed and heavily used machine learners — decision trees, support vector machines and neural nets — more or less mesmerised machine learning between 1980-2000. They initiated relatively novel and somewhat heterogeneous diagrammatic movements into data.

These diagrammatic movements, which we might characterise as *splitting*, and *marginalising* not only animate subsequent machine learners in producing newer techniques, they re-configure what counts as pattern. Since machine learning has no fixed idea of pattern (a term lacks much operational definition), then claims that machine learners uncover hidden patterns in data might be better grounded in the operational practices of working with differences.[2]

As machine learners of recent decades, the decision tree and support vector machine embody a new enunciative modality, a way of describing, locating and perceiving differences in which differences of degree and differences of kind are re-mapped. Every machine learner generates statements, but from different places, by somewhat different individuals, and from the different situations they 'occupy

2. *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'.

in relation to the various domains or groups of objects[2] (Foucault 1972, 52).[3] Practically, decision trees and support vector machines 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 ~~during 1980-2010~~ bristles with references to decision trees and support vector machine, as well as neural networks.[4]

3. ~~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.

4. 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).

Rather than seeing pattern as something discovered in data, the notion of enunciative modality suggests we should examine the diagrammatic operations that configure differences in the practice of machine learning, giving rise to a field of patterns attributed to objects or subject positions. ~~The~~ two machine learners that anchor this chapter are perhaps the most distinctive data mining, pattern recognition and predictive modelling achievements of the late ~~twentieth~~ century (at least judging by the citations and usage they attract). They differ greatly 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).

Splitting 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

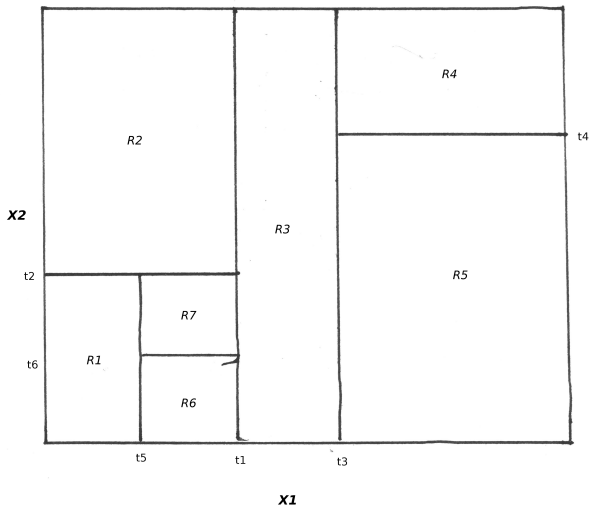


Figure 6.1: Recursive partitioning of the feature space

enunciative modality of the decision tree concerns 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 framing of differences in transforming. As *Elements of Statistical Learning* puts ~~it: 'tree-based~~ methods partition the feature space into a set of rectangles~~,~~ and 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 vector space of the data) can be mixed. Because the method cuts the vector space into a tiled surface (see figure 6.1), the features or data variables can be continuous or discontinuous. The ~~'~~simple models~~'~~ that tree methods construct each define one of the rectangular regions or partitions of the feature space. In ~~F~~igure 6.1, the different regions or partitions produced by a decision tree are labelled $R1, R2$ ~~etc~~.

Title	Year	Citations
The Achievement Motive And Economic Behavior	1964	20
Simplification Of Economic Models	1966	9
Data Dredging Procedures In Survey Analysis	1966	65
Advertising Performance As A Function Of Print Ad Characteristics	1967	32
World Affairs Information And Mass Media Exposure	1967	27
Juvenile Probation System Simulation For Research And Decision Making	1968	16
Presidential Elections Explanation Of Voting Defection	1969	12
An Interactive Technique For Analysis Of Multivariate Data	1969	9
Finding Variables That Work	1969	22
Brand Trial After A Credibility Change	1970	7

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

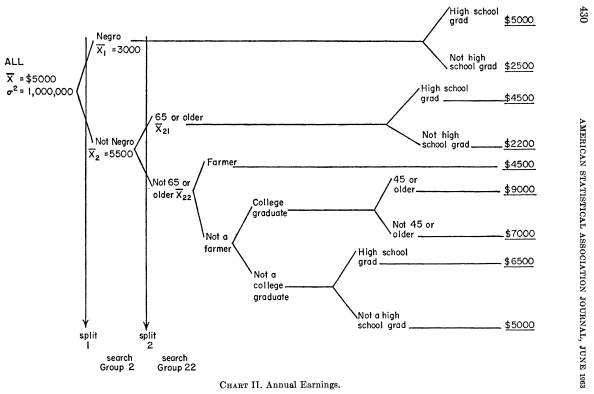


Figure 6.2: AID classifies annual earnings (Morgan & Sonquist, 1963, 430)

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 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 the initial work, `AID` was the object of trenchant criticism by statisticians and others, not for the classifications it used (see Figure 6.2), but for its 'pure' 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 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, 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).[5] Most of these criticisms can be seen as expressing conventional statistical caution in response to threats to validity, but they also address the core issues of pattern and difference: did the trees render differences arbitrarily?

5. 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.

1984: Differences in 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 in the following decade to investigate how decision trees work. Writing around 2000, Hastie, Tibshirani and Friedman, who could hardly by 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, `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

What happened between 1973 and 2013? Decision trees somehow stepped 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, computer 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).[6] This uneasy parallel effort between computer science and statistics still somewhat strains relations in machine learning today. 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

6. 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 20,00 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 that Breiman and co-authors.

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 supplanted 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, *Classification and Regression Trees*. As we have seen in previous chapters, classification and regression (predictive modelling using estimates of relations between variables) stage the two main sides of machine learning practice. Their concatenation with 'tree' attests to a renovation of existing machine learning approaches behind a single facade.

The implementation of machine learning techniques in `R` accentuates the statistical side of decision tree practice, but that has certain forensic virtues not offered by commercial or closed-source software often produced by computer scientists. 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 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 underlying algorithmic process:

recursive partitioning.[7]

Listing 6.1: Decision tree for `iris` dataset

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

R.A. Fisher's `iris` dataset, which contains 150 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).[8] 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 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). Classification in decision trees operates by splitting each of the dimensions of vector space into two parts (as we saw in figure 6.1). These splits institute branches along which differences are hierarchically ordered in a tree structure. The recursive splitting algorithm draws a diagram of hierarchical differences. The problem here is that many splits are possible. What is a good split or ordering of differences?

> The first problem in tree construction is how to use $\mathscr{L}$ to determine the binary splits of $\mathscr{X}$ into smaller pieces. The fundamental idea is to select each split of a subset so that the data in each of the descendant

7. 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.

8. `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.

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 to what extent 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 classifiers such as the perceptron or linear regression model, recursive partitioning combined with measures of node impurity transforms data by cuts or divides. Whereas in linear model-based machine learners, the intuition motivating the function-finding or learning was 'find the line that best expresses the distribution of the data', here the intuition is more like: 'find the cuts that minimize mixing'. Good splits decrease the level of impurity in the tree. In a tree with maximum purity, each terminal node — the nodes at the base of the tree — would contain a single class.

In Figure 6.3, 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. 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 2.45, then the iris is *setosa*. 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. A decision tree has little

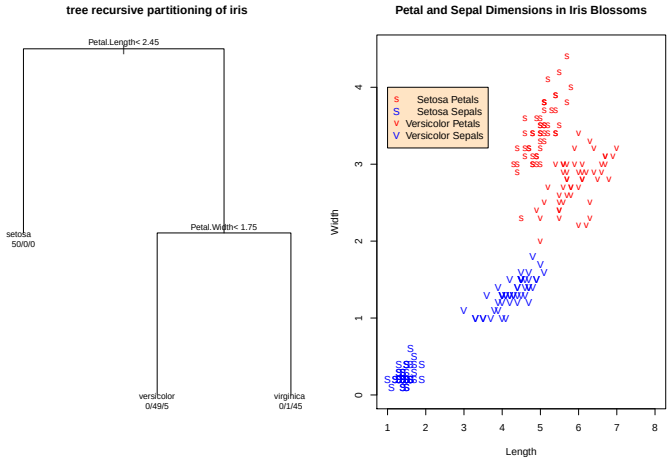


Figure 6.3: Decision tree on `iris` dataset

trouble ordering differences between species of iris.

Like logistic regression models, neural networks, support vector machines or any other machine learning technique, decision trees order differences in terms of specific qualities and logics. Recursive partitioning split and sub-divide the vector space to capture every minor difference between cases, and thereby achieve a ever-closer fit to the individual or sub-individual variations. Although the partitioning or splitting rules have strong statistical 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 identifying differences. During construction of a decision tree, recursive partitioning splits features in the data into smaller and smaller groups. 'The goodness of the split', wrote Breiman and co-authors, 'is defined to be the decrease in impurity' (Breiman et al. 1984, 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 differences of the individual case to such a degree that it could end up seeing categorical differences

everywhere. Operating 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 sensitive to differences and generalize poorly (that is, they suffer from generalization error \index{error!generalization}). 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 may have low bias but high variance.

Limiting differences

Given this problem of unstable difference, much of the development of decision tree did not revolve around how to construct them, but how to limit their growth so as to manage tensions between pure but unstable differences and impure but stable classification. As *Elements of Statistical Learning* puts the problem in its 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 optimises the decision tree as a

machine learner. 'Cost complexity pruning' extends the optimisation we have already discussed in relation to linear regression and logistic regression models (in chapter 4). 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 — iteratively observes and tests different versions of tree against each other. 'We define the cost complexity criterion,' write Hastie and co-authors, as:

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

The idea is 'to find, for each α, the subtree $T_\alpha \subseteq T_0$ to minimize $C_\alpha(T)$' (Hastie, Tibshirani, and Friedman 2009, 308). For present purposes, we need only recognise that the cost complexity function re-configures a large decision tree (T_0 in equation 6.1) by cutting or pruning it back through optimisation that balances between the complexity of the tree and its stability. Tree construction is as an optimisation problem, in which the variation of a parameter (α) allows minimization of a derived value (the cost C_α).

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 (312). The very diagrammatic form that allows decision trees to be observed and interpreted is also the source of their instability.

Regardless of this instability, the diagrammatic composition of the tree through splitting and pruning negotiates between two different ways of doing difference.

The shift from `AID` to `CART` enunciates a change in how patterns of difference become visible. The decision tree algorithm superimposes recursive partitioning and cost-complexity pruning to configure a mode of enunciation of differences. It creates a new rules of differentiation of individuals, facts, things and relations.[^5.101] Differences in a decision tree — the combination of purity and density that comes from recursive partitioning and cost-complexity ~~pruning — re-configure~~ what counts as pattern.

Decision trees have been heavily used in credit risk assessment as well as many biomedical models. Does their popularity stem from the legibility of the statements they produce, even if those patterns prove unstable? Or is the success of the decision tree perhaps better understood as a change in the differentiation of patterns more generally, their mode of enunciation, in which case, decision trees would only be one instance among many? If we understand machine learners as generating populations of statements, the transformation and re-modelling of the decision tree as classification and regression trees suggests a subtle, non-localizable discontinuity. The later development of the decision tree and its subsequent transmogrification into random forests (Breiman 2001b), that grow a myriad of small decision trees disperses kaleidoscopic fragments of classificatory order with only partial or provisional stabilisation in visible pattern. In such developments — and we could also consider here techniques, models and methods of ~~'boosting,' 'bagging,' or the 'ensemble learning' that conducts 'supervised search in a high-dimensional space of weak learners'~~ (Hastie, Tibshirani, and Friedman 2009, 603), pattern has an

operational rather than visible mode of togetherness.

The successful dispersion of the support vector machine

In growing and pruning decision trees, and even more markedly in support vector machine, patterns play out in dispersion and discontinuity rather than in regular geometry. While machine learners order differences, that ordering becomes increasingly difficult to see as it is dispersed. Take the case 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 appear 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 algorithm.[9]

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 6.2. 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

9. The support vector machine is distinctive in its transformations of data, and this owes something to history, politics and geography. 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 under-

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 6.2: Most cited papers on support vector machines

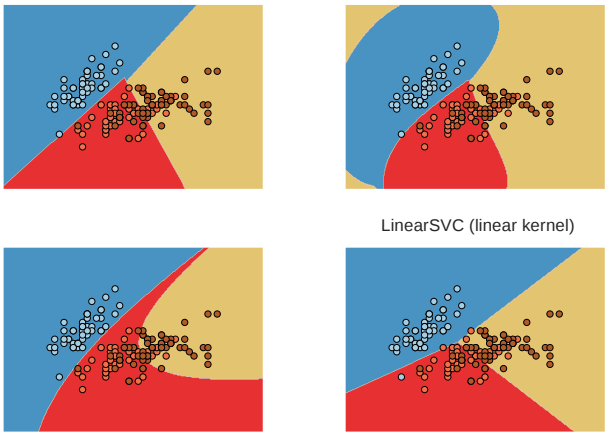


Figure 6.4: Support vector machine on `iris` dataset

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 near the top-cited papers.[10] This kind of growth betokens high levels of interest, identification and investment on the part of the researchers~~,~~ and presumably more widely.

I suggested ~~above~~ that the classification tree (and then random forests) illustrate an enunciative modality anchored in a tension between recursively partitioned differences and classificatory stability.

10. *Elements of Statistical Learning* also devotes a chapter to support vector machines (Hastie, Tibshirani, and Friedman 2009, ~~C~~hapter 14).

The support vector machine shown in Figure 6.4 demonstrates a different change in what counts as pattern. The decision boundaries in the sub-graphs have different contours, contours that suggest a more mobile construction. While the name 'support vector machine' 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. Fisher developed the 'first pattern recognition algorithm' (Cortes and Vapnik 1995, 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 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 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 vector space into regions of constant classification. These long-standing discriminant functions were reconstructed during the 1990s in the form of the support vector machine, giving rise to new statements about differences, statements that can be glimpsed in table 6.2 in the range of things, facts and

beings running through the titles of the papers.

Differences blur?

Decision boundaries change in two ways in support vector machines. They blur and bend, again affecting what counts as pattern. 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 transforms 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 (`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.[11]

In their experiments with digit recognition (shown in figure 6.5,

11. 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 (Hastie,

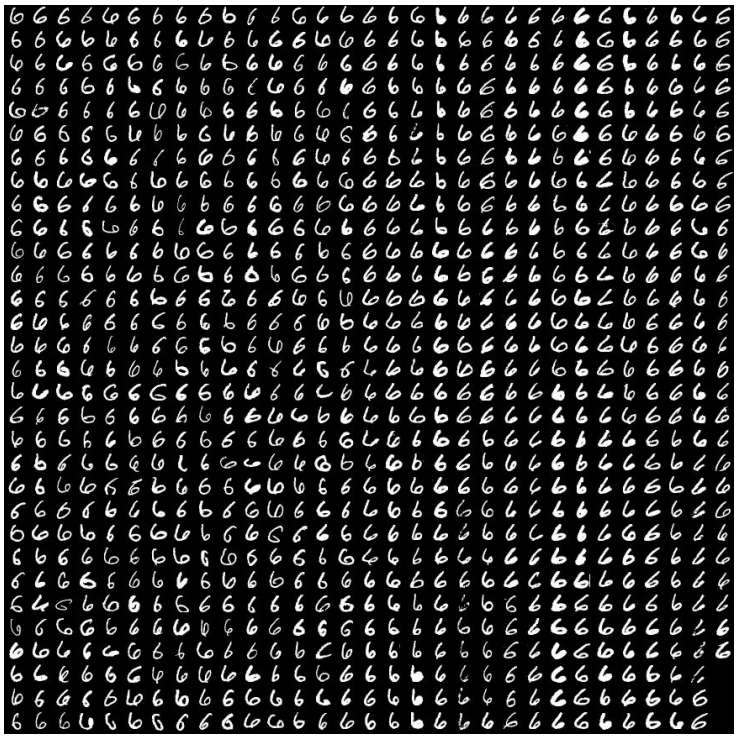


Figure 6.5: MNIST ~~P~~ostal ~~D~~igits

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 dimensionality. Support vector machines deal with blurred differences or continuous variations by superimposing two operations: 'soft margins' and 'kernelisation.' 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 develops Fisher's linear discriminant analysis ~~since~~ it searches for a separating hyperplane in the data. ~~While~~ linear discriminant analysis constructs a hyperplane by finding the most likely linear 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 differentiating classes will run near to the cases — the *support vectors* — that are most difficult to classify. Hard-to-classify cases become the 'support

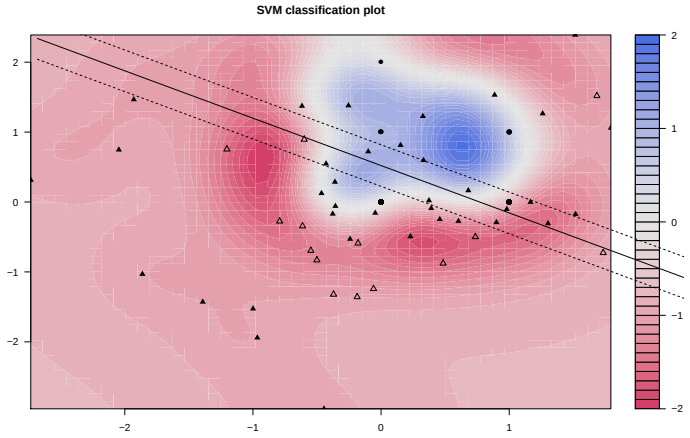


Figure 6.6: Margins in a support vector machine. The margins ignore most of the data and only focus on the hard-to-classify cases

vectors', whose relative proximities tilt the decision surface in various directions. In contra-distinction to the $N = \forall \boldsymbol{X}$ proposition (discussed in chapter 5), the machine learner discards much of the data. In contrast to linear discriminant analysis, as Ethem 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 6.6 has appeared in many slight variations in the last two decades. 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 6.6, 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 accommodating inseparable or overlapping classes. This is something that other machine learners (~~for instance~~, the perceptron) cannot do.

~~While~~ the geometrical intuition here is that some data points (cases or observations) will 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 ~~F~~igure 6.6 calculated? Locating the optimal separating hyperplane and a limited number of permitted mis-classifications presents a complicated optimisation problem. As *Elements of Statistical Learning*, following Cortes and Vapnik's formulations, formalizes it, the problem can be stated in terms of minimization:

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

(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 ξ_i (420). In this complicated optimisation problem (one that is difficult to understand without extensive mathematical background), familiar elements include the parameters β in the linear form of $x_i^T \beta + \beta_0$, which is the equation defining the hyperplane, as well as triple operations of addition ($\sum$) of all the values ξ_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 mathematical functions, their diagrammatic relations, and the way in which they contain both the generalizing regularities (algebraic icons and indexes, the linear equation, the

repeated summation of all values, the shaping parameters) and forms of variation (the presence of the misclassification measures ξ_i) should focus 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 inseparable or overlapping classes, and hence with blurred patterns of difference. Given that support vector machine permits instances that lie on the wrong side of the separating hyperplane, irregular differences no longer function as errors (as they would appear in most linear classifiers such as linear discriminant analysis and logistic regression), but as elements in 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 steering 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 in the machine consists in finding a way of transforming data into differences according to constraints.

Bending the decision boundary

The support vector machine reinstates a linear decision boundary as the enunciative mode of difference. Yet it transforms that boundary. In the abstract of their 1995 paper, Cortes and Vapnik briefly describe the how the support vector machine revises 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)

Another example of vector space transformation (discussed in chapter 3), 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 located amidst a non-linear mapping of the data.[12] Cortes and Vapnik's support vector machine constructs a new domain - 'a very high dimension feature space' - where inseparable differences start to disentangle themselves. The constructed dimensions do 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 6.4). 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.[13] Research in machine learning, whether it has been primarily statistical, mathematical or computational, counte-

12. As we have seen on several occasions, the 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.

13. 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

nances 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 6.3:

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

(Hastie, Tibshirani, and Friedman 2009, 423)

In equation 6.3, a re-mapping of equation ?? occurs particularly through the substitution of a product $\langle h(x_i), h(x_i^{'}) \rangle$ for x. All of the data x ~~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. (Breiman 2001a, 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 generation of differences 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 limited range of movements prescribed by the kernel functions (polynomial, radial, etc.).

Elements of Statistical Learning puts 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 vector space (see chapter 3) 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*.

Top 10 Algorithms for Data Mining (Wu et al. 2008), a widely cited computer science account of data mining, justifies the operation in relation to entangled differences:

> 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'.[14]

What does the transformed feature space combined with the computational short-cut of the inner product do in practice? Describing 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 in classifying the handwritten numbers of the `mnist` data. They recount 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 $1x10^{16}$ dimensions), the error rate drops to just over 4%, close to the errors made by 'human performance' (2.5%) (288).

14. This cheapness appeared already in the cat machine learner discussed in the introduction. Heather McArthur's `kittydar` cat image classifier implemented a support vector machine in Javasript that runs in a browser. Cats are classified cheaply there.

Instituting patterns

The engineered movement of various machine learners do not simply discover differences. They construct, identify and optimise

distributions or patterns of difference. They do it in different ways. Sometimes they take for granted the ~~very~~ possibility of identifying differences in data, as if all differences must be visible and divisible given the right partition. At other times, intrinsic inseparability is taken into account as part of the pattern. 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 differences from decision trees and their development into random forests, or from linear discriminant analysis and its re-formalization as the support vector machine in some of their operational specificities? First, patterns are multiple in machine learning. We could also have tracked the movement between the perceptron (Rosenblatt 1958) and the 'deep learning' convolutional neural networks (Hinton and Salakhutdinov 2006) of more recent practice (see chapter 8). Second, each of these shifts bears witness, I have been suggesting, to the emergence of a new enunciative mode that disperses patterns as the visible form of difference into a less visible but nevertheless operational space. A single decision tree becomes thousands in a random forest. A relatively small number of dimensions in the 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 amid~~st~~ the massive normalized accumulations of event.

What counts as pattern today? The visually interpretable shape

of a decision tree cascades into the statistically observable trade-offs between fine-grained classification and cost-complexity considerations, between recursive differentiation and general sparsity. 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.

What does it matter if pattern disperses into operations? From the standpoint of critical thought, it might be that learning to find dispersed patterns only intensifies a tendency 'to see differences in degree where there are differences in kind' (Deleuze 1988a, 21) It would, however, be relatively pointless to assert primacy of differences in kind. A more constructive and experimental challenge lies in exploring differences in kind within the computed differences of degree active in machine learning. Despite their quite different ways of partitioning, separating or propagating differences, support vector machines and decision trees define possibilities of grouping and spacing differences, sometimes through purifying, sometimes through bending and blurring, and sometimes through multiplying. These groupings and spacings attract, generate and accumulate propositions. This grouping-spacing, 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 tunable soft margins and kernel functions 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).

7

Regularizing and materializing objects

> 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. (Deleuze 1988b, 19)

What is the materiality of machine learning? The opening pages of *Elements of Statistical Learning* present four somewhat excessive objects — spam email, handwritten digits, prostate cancer, and 'DNA Expression Microarrays' — and list six 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 ply them? Although machine learning occurs in the thick of a control crisis (Beniger 1986) (as suggested in chapter 1), I will suggest here machine learning also occasions viscous multi-temporal and inter-objectively distributed enactments of things such as financial markets, media platforms, chronic diseases and living things. These are all hyperobjects that epistemically, infrastructurally, economically and socially individuate through machine learning.

The last of the vignettes, the DNA microarray, comes from the life

sciences. It attracts a whole page colour figure — a heatmap (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 abruptly reappear in a discussion 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 extensively, in a final chapter (Chapter 18) new to the second edition of the book on 'High Dimensional Problems'. Apart from one passage where the Hastie and co-authors develop a document classifier for their own 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.

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

Operational formations usually encompass scientific fields. Alongside the operational problems of spam email filtering, image or handwriting recognition, scientific research into biological processes constitutes a major reference point and, I will suggest, an axis of materialisation for machine learning. In the archaeology of its operational formation, we could say that the scientific domain of genomics has a strongly referential effect on machine learning. What is a referential? For Foucault, a referential '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). For Foucault,

a referential forms an integral part of the enunciative function, the mapping of sites, subject positions, enunciative modalities, forms of accumulation and differentiation at work in the production of statements.

Why does the referential of machine learning matter? When hyperobjects are machine learned, they are re-constituted (in vector space, as optimisation problems, as probability distributions and patterns of difference). Conversely, as I will suggest in this chapter, they become a site of materialization, cross-validation and regularization for machine learning in its production of knowledge. But that referential status, which authorises and imbues statements with value, comes at a cost. The plurality or multiplicity of the hyperobject — genomes, stock prices, etc. — will be regularized and ranked, re-used and transcribed by machine learners over and over to lend coherence to the operational formation and its system of statements.

Genomic referentiality and materiality

```
gaagctccac accagccatt acaaccctgc caatctcaag cacctgcctc
tacaggtacc
```
(NCBI 2016)

By contrast with industry, commerce, media and government, where much that happens is obscured from view, the great virtue or genomic science is the relative openness of its workings and its resolute insistence on DNA as the primary form of data. The fact that data practices are relatively generic and accessible means that critical research into transformations associated with genomic data and knowledge can accompany nearly every aspect of practice.

Genomic data exhibits some specific features. The first concerns what I earlier called data strain. Genome data, a tiny fragment of which is shown above, inflates the vector space. Genomics generates

new versions of the now familiar problems of data dimensionality. The abundance, diffusion, heterogeneity or impaction of genomic data thwarts its examination, tabulation, and regulated circulation. Genomics data also presents unusual ratios 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 $\forall \boldsymbol{X}$ 'Law of Large Numbers' (Hacking 1990, 99–104), which holds that the accuracy of statistics tends to increase with more observations.

Since the early ~~nineteenth~~ century, biology and cognate disciplines have sought to explore problems of time, genesis, duration, activity and process in a ~~very~~ broad spectrum of living things. Contemporary genomics seeks to elicit, as many commentators have noted, knowledge of biological, evolutionary, biomedical and environmental processes from the long DNA sequences comprising genomes. The primary 'object' in genomics is a $\forall \boldsymbol{X}$ data form, the 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 a dataset comprising the complete sequence of DNA potentially re-bases knowledge of many different biological processes, ranging from evolution (phylogeny), development (ontogeny), metabolism, structure and pathology. If nothing else, genome comparison promises 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]

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

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 biological significance (for instance, evolution from a common ancestor).

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]

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 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

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 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 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).

the inseparability of distinct variations' (Deleuze and Guattari 1994, 21).

Whatever knowledge subsequently derives from a genome (genes, mutations, evolutionary relationships, variations associated with disease, heredity or individual identity), genomic data and hence the genome itself as a scientific hyperobject is deeply probabilistic. From assembly onwards, through the ancestral probabilisation embodied in heavily-used biological databases, the indelible errors, the entangled reliances on accumulated biological knowledges make genomes a particularly challenging site of machine learning activity.

Elements of Statistical Learning'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 tightly coupled referential for machine learning even if relatively few machine learners have, to date, managed to work with whole genome sequence data. The relatively long-established referential entanglement (at least 25 years, and perhaps more) of genomics and machine learning is strategically important in the generalization of machine learning, in the processes whereby 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.

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 the begin to appear in digital signal processing for wireless communication and video image compression (Mackenzie 2010) and above all, in speech recognition (Rabiner 1989).

	Frequency	Discipline
1	1301	computer science, artificial intelligence
2	923	engineering, electrical & electronic
3	520	statistics & probability
4	401	computer science, information systems
5	344	computer science, interdisciplinary applications
6	332	biochemistry & molecular biology
7	284	mathematical & computational biology
8	260	biochemical research methods
9	259	biotechnology & applied microbiology
10	227	neurosciences
11	218	computer science, theory & methods
12	198	radiology, nuclear medicine & medical imaging
13	197	multidisciplinary sciences
14	189	genetics & heredity
15	181	immunology
16	157	ecology
17	155	imaging science & photographic technology
18	125	automation & control systems
19	122	engineering, biomedical
20	106	computer science, hardware & architecture

Table 7.1: The top 20 disciplines of the top 5000 cited research publications in machine learning, 1990–2015

The genome as threshold object

During 1990–2015, biology, and particularly molecular and then genomic biology, has a very high visibility in the machine learning research literature. (See table 7.1.) 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, come into play, are potentialized and regulated, and take on promissory epistemic value as zones of collective individuation through 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, the genomic referential marks a threshold of materialization. The archaeological approach to materiality is somewhat unusual. Given his interest in the formation of statements, Foucault understands materiality as a regulatory process operating in an enunciative function. Foucault writes:

> 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).

In the archaeology of an operational formation, locating specific practices, places and times of reinscription, transcription, and possibilities of reuse carries more weight ~~more~~ than any direct conceptual account of materiality. What would materiality in this sense mean for machine learning?

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 cross-validation of machine learning as a life-death relevant knowledge practice. Genomes first of all authorize infrastructural vectorizations 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 7.1 (Krzywinski et al. 2009)), the speaker, Urs Hölzle, Senior Vice President of Infrastructure at Google

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.

'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 cancer genomics. Their interest was steered toward the immediate availability of computing power: from 10,000 to 600,000 cores in a few seconds.

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.

Such drastic infrastructural re-scaling attests to the provocation of the genomic referential. The Google Compute demonstration is, I would suggest, typical of how genomes, genes, proteins and biological sciences more generally, authorize differentiation of individuals, events and things through machine learning. This differentiation 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. Google's press release emphasises the distribution of learning across an infrastructure:

> 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).

Google re-purposes an algorithm developed by engineers at Intel Corporation and Amazon, and draws on genomic datasets provided by the Institute of Systems Biology, Seattle, a doyen of big-data genomics. The demonstration animates the `RF-ACE` (a further development of Breiman's random forests discussed in chapter 6) by re-drawing 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 pro-

cess. This confluence of commerce (Amazon), industry (Intel), media (Google) and genomic science (ISB) exemplifies the re-inscriptive or transcriptive materiality of machine learning.

Genomic knowledges and their datasets

In the infrastructural materiality of these demonstrations and examples, whether they come from *Elements of Statistical Learning* or from Google Compute Engine, the object of knowledge — genomes, genes, proteins — does not figure in terms of its original discipline or scientific field (typically cancer biology). The scaled-up demonstration of `RF-ACE` on Google Compute assembles only a general system of references ~~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. 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 recogni-

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 escribe as 'becoming answer to statistical requirements' I would

tion, 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 biology) fit? Almost all of the major machine learners, albeit supervised or 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 advent of 'wide, dirty and mixed' data

We can see this referential cross-validation at work in the shape of genomic data. The DNA microarray data extensively modelled in the final chapter of *Elements of Statistical Learning* highlights some elementary problems of shape associated with genomic data. The `iris` dataset (R. A. Fisher 1936), perhaps the most heavily used pedagogical dataset in the literature, 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 7.2: 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 table 7.2 to infer the value of the `Species` response 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 biological procedure. Even in the excerpt shown in Table 7.2, 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). `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 statistical 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 an irregular texture, which, I would suggest, we should try to grasp if we want to understand how genomic data constitutes a complex volume 'in which heterogeneous regions are superposed' (Foucault 1972, 128). Clues to the irregularity of genomic data 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 (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, genomic data as a field or emergence and differentiation in the production of statements matters. 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), precisely in order to address the diagnostic problem of distinguishing different tumors types without resort to new experiments or biological knowledge.[8]

GENE1	GENE2	GENE3	GENE4	GENE5	GENE6	GENE7	GENE8	GENE9	GENE10	GENE11	GENE12	GENE13	GENE14	GENE1
0.77	-2.44	-0.48	-2.72	-1.22	0.83	1.34	0.06	0.13	0.57	1.50	0.39	1.63	0.82	0.0
-0.08	-2.42	0.41	-2.83	-0.63	0.05	1.43	-0.12	0.46	0.16	1.15	0.38	1.56	0.01	0.1
-0.08	-1.65	-0.24	-2.88	-0.89	-0.03	1.16	0.02	0.19	0.50	1.39	-0.53	1.61	-0.21	0.0
0.97	-2.38	0.63	-1.74	-0.85	0.95	1.09	0.82	-0.28	0.99	1.01	0.08	1.05	0.97	-0.1
0.08	-1.73	0.85	0.27	-1.84	0.33	1.25	0.77	0.03	0.28	1.11	-0.39	1.19	0.42	-0.3

Table 7.3: Small round blue-cell tumour data sample (Khan, 2001)

The sample of the `SRBCT` data shown in table ?? does not readily accommodate the width of the 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 number of biological samples but a huge amount of sequencer or microarray data for each sample. Much genomic data shares this generic feature of width.[9]

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 (673)

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

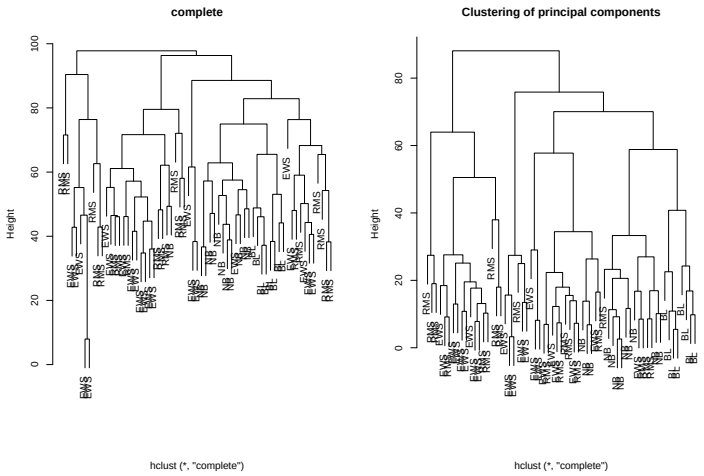


Figure 7.2: Hierarchical clustering of the SBRCT gene expression data

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 shown in figure 7.2.) Even the identification of the several thousand genes whose levels of expression are measured by the microarray experiments presupposes much preceding work on DNA sequences and the identification of protein-coding DNA regions 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 bare base DNA sequences might suggest.

Cross-validating machine learning in genomics

The linear sequences of DNA data mix and diffuse partly through the archival accumulation that allows them to be superimposed, annotated and layered, but also through the many efforts to traverse their expanded volume using classifiers and predictive models. A

recent review in the journal *Genomics* highlights the increasing bearing of machine learning techniques on genomic science:

> 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).

Commentary on the 'highly correlated structure,' not just the volume, of genomic data, points to another referential operation concerning the differentiation of things. Many such statements highlight the incompatibility between a surging multiplicity of data forms and the constraints of existing statistical modelling techniques ('standard statistical models'). So for instance, ~~Chen and co-authors~~ recommend the use of the random forest (RF) 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, … including prediction and classification, variable selection, pathway analysis, genetic association and epistasis detection, and unsupervised learning (323)

Familiar machine learning vectorisation keywords such as 'large p, small n' and 'high dimensional' pepper their recommendations. But the key terms on the genomics side of this formulation would perhaps be 'pathway analysis', 'genetic association,' and 'epistasis.' Such biological terms point to forms of relationality 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 brought shifting patterns of gene expression to light. 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.

How does machine learning differ from the statistical practice that has underpinned much of modern biology? The analysis of `SRBCT` gene expression in *Elements of Statistical Learning* is symptomatic of a mutual articulation, a cross-validation that entangles genomics and machine learning. The overt arrival of machine learning techniques in genomic research was initially largely concerned with the problem of variations in gene expression (~~-~~and in fact~~,~~ nearly all of the analysis of genomic data in *Elements of Statistical Learning* explicitly deals with various cases of gene expression). On the one hand, the genomics data promises legibility 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.

Compared ~~to~~ the refined algorithmic craft of whole genome assembly (Venter et al. 2001; Myers et al. 2000; Pevzner, Tang, and Waterman 2001), the handling of the problem of gene expression 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 prob-

lem of learning to classify differences in disease or other health-related conditions. 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.

Like Chen, *Elements of Statistical Learning* begins by addressing the problem of the shape of the data. '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). What is this regularization? Like the re-distribution of classification into a randomised population of machine learners (see chapter 5), regularization governs a ~~n~~ potentially unruly plurality through a form of training and 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 regularization as a technique 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 genomic and other referentials (transactions, social media, etc.) displays some of the traits — mas-

siveness, transience, plurality — that Foucault identifies as key targets of regulation for the operations of disciplinary power focused on the social body or populations. The 'target,' a term often used in machine learning to describe the type, group or response being modelled, in genomics is often subtle variations (in gene expression, in phylogeny, in pathogenesis), and these variations are widely dispersed in genomic sequence data and in the populations it stems from. Foucault's account of supervision (*surveiller*) and penalisation as disciplinary techniques responding to 'popular illegality' (Foucault 1977, 130) dwells on 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 target a more subtly distributed nexus of relations in disciplinary power. Often understood in terms of enclosure and surveillance, disciplinary power, according to Foucault, operates through ranking: '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 regularization deals with the problem of too many variables in the microarray datasets. 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, support vector classifier—essentially the same ordering movement occurs. Regularization re-scales the excessive potential relations of 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 ranked by importance, and then 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.

A technique called 'lasso regression' displays features that might help us grasp how machine learners regularize genomic data. Remember that the linear regression model with its diagonal line or plane running through vector space provides the underlying intuition for many machine learners. We have seen the function in Equation 7.1 several times already in different variations, including logistic regression used for classification of types or groupings.

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

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 p columns of the microarray dataset. The model tests how different patterns of gene expression associate 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.

The regularized version of the linear regression framework known as 'lasso'—Least Absolute Shrinkage and Selection Operator—introduces a different form of training and observation of model construction. This train hinges on the lasso penalty shown in equation 7.2[10]

$$\widehat{\beta}^{lasso} = argmin_{\beta} \left\{ \frac{1}{2} \sum_{i}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}. \quad (7.2)$$

(Hastie, Tibshirani, and Friedman 2009, 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 $argmin_{\beta}$ suggests that the algorithm should optimise the set of values for β that minimize the overall value of the function. It balances the costs of reducing the sum of residual errors shown in the first half of the equation, and minimizing the sum of the absolute values of the model parameters β_j in the second part of the function. The optimizing double movement re-shapes its expression of the data along a diagonal line drawn as the algorithm gradually introduces and scales all of the features in the vector space **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. (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.

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.

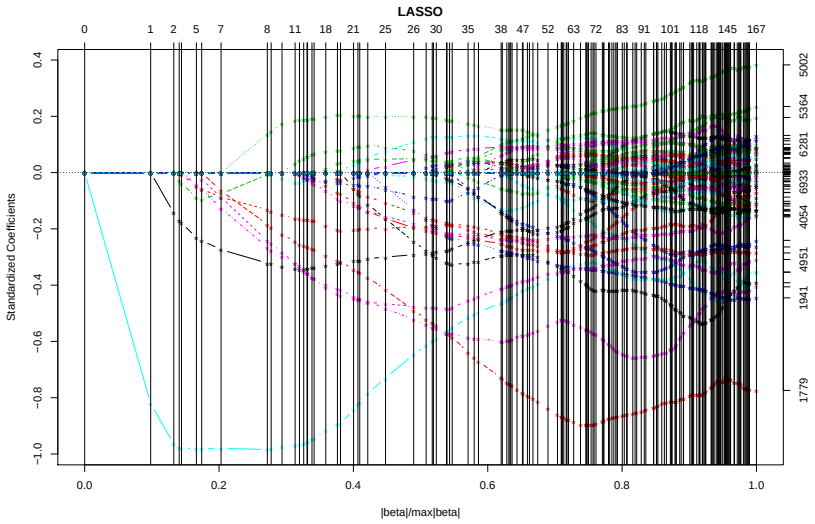


Figure 7.3: Shrinkage path of coefficients for Lasso regression on (Golub 1999) leukemia data

)

Regularization 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. 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 transient multiplicities (such as gene

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

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.

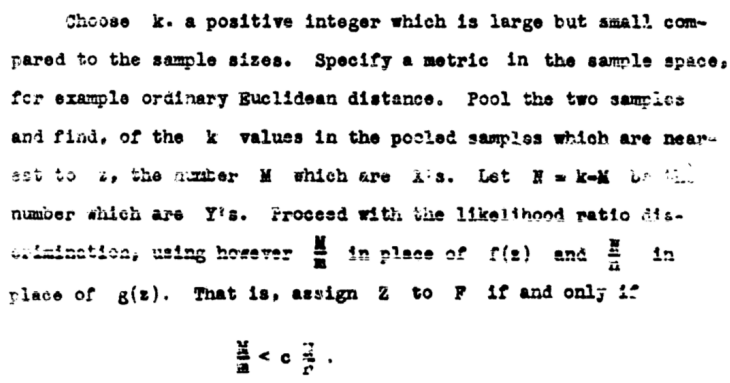
In the shift from MAQC-I to MAQC-II, the problem of variations in the predictions produced by the machine learning models moved to center-stage. The problem of variation arises not because any of the different modelling strategies used in machine learning gene expression datasets are wrong or erroneous, but because every model transforms the 'feature space' (Parry et al. 2010, 292) in a different way (as we saw in chapter 6 in discussions of different treatments of dimensionality). In the MAQC-II consortium, 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 MAQC 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 population of machine learners show variations.

What accounts for this variation? First of all, the teams did not build single models. As is the norm in machine learning, they iterated over thousands. In their attempt to normalise the variations of their models, one of the research groups in MAQC-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). A striking proliferation of models on a population-scale strives 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. This population exhibits many of the problems of variation, irregularity, transience and plurality found in the genomic referential itself.

Variations in the object or in the machine learner?

'The method of k-nearest neighbors makes very mild structural assumptions: its predictions are often accurate but can be unstable'



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 $\frac{M}{m}$ in place of f(z) and $\frac{N}{n}$ in place of g(z). That is, assign Z to F if and only if

$$\frac{M}{m} < c \frac{N}{n} .$$

Figure 7.4: The earliest formulation of the *k*-nearest neighbours model from Evelyn Fix and Joseph Hodges' work [a]

write Hastie and co-authors (Hastie, Tibshirani, and Friedman 2009, 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 entire algorithm:

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

'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 N_k, and uses that value to predict the result (a classification or a prediction) for a given point or instance. 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 transforming the vector space, is that proximity in vector space implies similarity in class or grouping. 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

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 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?' (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.

to have only one parameter, the value k, the number of neighbours that a given model includes in its definition of a 'neighbourhood.' In contrast to the linear forms of the models (formulated in equations 7.2 or 7.1), equation 7.3 seems to require little training, supervision or regularization to work as a classifier. While nearly all of the models discussed in this and earlier chapters work with a smooth functional form of the line or curve as their basic way of transforming vector space, k-nearest neighbours generates highly non-linear boundaries wending their way through the data. Because they are not guided by parameters (apart from the value of the hyper-parameter k), these boundaries can be unstable.

Even when data belongs to two classes (e.g., `normal` vs. `not-normal`), decision boundaries produced by k-nn can be unstable. The example in figure ?? shows two models, one for $k = 5$ and the other for $k = 2$. Each model examines the relations between 2, 5 points in deciding whether a particular case belongs to one class or another. While *k-nn* constructs local clusters and traces out an irregular decision boundary, this classificatory power comes at the cost of instability. (This is another version of the bias-variance decomposition of machine learner errors discussed in chapter 5.)

More data, or wider data exacerbates the instability. As dimensions or features in the dataset increase, the local neighbourhood needed to capture a fraction of the volume of the data expands. It becomes 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-nn* allows for non-linear interactions between features, for instance, small differences in the number of

points in particular neighbourhoods can drastically affect some stretches of the boundaries (as we see in comparing the right and left hand plots in figure ??). 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 begin to see how a MAQC-II team might have produced 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

Cores, microarrays, SNPS and many models: infrastructural scaling, biological variation and the populations of unruly machine learners entwine in referential entanglement. If genomes are scientific hyperobjects (with epistemic, speculative, financial and biopolitical resonances), what part does their referential cros-validation with machine learning play in the transformation of knowledge?

Genomic data — beginning with DNA sequences, then levels of gene expression, followed by genome-wide association studies of small mutations — has been a constant $p \gg N$ antigen in machine learning. Techniques of regularization — the lasso — of linear models discussed in this chapter came to light, and were first demonstrated on 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 cross-validating in practice. Scientists, statisticians, datasets and machine learners 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. In 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 a referential flow of materializing transcription, infrastructural cross-validation, and regularizing differentiation. An entity whose constitution is thoroughly dependent on prediction or algorithmic classification displays variations and grouping (such as gene expression, the linkage disequilibrium of SNPs, the seeming abundance of junk DNA that is actually functional, etc.) that attract further efforts to differentiate, regularize, and classify ever more subtly distributed differences. Elementary practices in contemporary genomics such as sequence alignment were explicitly formulated as generative models to be constructed using algorithms such as expectation maximization. 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, machine learning establishes a new set of conditions for the exercise of scientific research, and configures new kinds of statements, new types 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).

What is at stake in approaching machine learning in a scientific setting like genomics? Foucault writes that 'we should distinguish carefully between *scientific domains* and *archaeological territories*' (Foucault 1972, 183). Knowledge stems from the practices that

connects objects, field, subjects, statements, and institutions. Sciences are always localized within a field of knowledge that may exceed, and mutate in ways that alter, local sciences. Science, for Foucault and perhaps for science studies more generally, needs knowledge practices that exceed, surround and indeed do something different to science. Machine learning is just such an operational formation.

Could we pose or address any normative questions by becoming aware of and articulating machine learning with science 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*.

I see matters as slightly more complicated than an instrumental production of knowledge. In the biosciences of the last two decades, machine learning 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 putatively linear, monolithic and searchable DNA sequence data. 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 spatial re-shaping in life-forms remains. None of the examples of genomic data in *Elements of Statistical Learning* use whole genome. 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, 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 MAQC-II.

The enunciative function of machine learning inscribes the possibility of genomes as multi-temporal, inter-connected expressions of variation. Such regularizing and potentializing of things on new infrastructural, collective and domain-specific scales outstrips instrumental purposes. In *The Archaeology of Knowledge*, Foucault describes discourse as 'controlled, selected, organised and redistributed according to a certain number of procedures, whose role is to avert its powers and its dangers, to cope with chance events, to evade its ponderous, awesome materiality' (Foucault 1972, 216). Something similar flows through operational formations such as machine learning in their entanglements with sciences. Controlling, selecting and organizing, it almost inadvertently affirms a ponderous, 'awesome' materiality of data practice.

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 (Foucault 1972, 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)

Who is the subject of machine learning? 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 software was to manage 'knowledge' in call ~~centres~~ such that any query from a caller could be readily answered by staff who would query a knowledge management system to find answers to the query. This system was marketed on the promise of machine learning. It relied on 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. 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. The rest of the team 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 seen with `kittydar`, the position of machine learners is changing. They are no longer exotic or specialized, but banal or occasionally spectacular. Hilary Mason, who was Chief Scientist at bitly.com (an online service that shortens URLs), outlined an everyday 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. (Hilary Mason, Chief Scientist, bitly.com) (*Hilary Mason - Machine Learning for Hackers* 2012)

These 'things that are different,' what I have been calling an operational formation, assign subject positions. In what ways does machine learning assign subject positions? 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 could become in relation to expansive changes occurring around and in their work. 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.'

Neural nets stand at an intersection of infrastructure and cognition, and then propagate subject positions forwards and backwards. Their operations encourage and elicit competitively ranking as an ordering that not only compares human and machines, but subject positions more generally.

Propagation across human-machine boundaries

The concatenation of 'one package, one mind' does not definitively allocate agency to people or things. (A 'package' after all is another name for a library of code.) Mason adumbrates the outline of a subject position 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

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). 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

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 exciting time to be in any of these things') and assign them a marginal albeit still pivotal position in relation to privileged machines?

Machine learner subject positions are the topic of this chapter. I focus on artificial neural networks, or neural nets, in their various forms ranging from the multilayer perceptron (MLP) to the convolutional neural nets (CNN), recurrent neural nets (RNN) and deep belief 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

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.

puts it), re-drawing human-machine differences, sometimes making it hard to seen what subject position they entail, where subjects are located or what they say, see and do.

Like other machine learners, neural nets re-draw human-machine differences. 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, model parameters, and error rates, 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,' they comment, '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). 'Looking,' as often the case in machine learning, takes the form of diagramming a pattern, partition or strain in the data.

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.

The substitution of 'mind' and model reproduces many aspects of the figure 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) suggests a rather different subject position. Archaeologically, the description of subject positions entails more than giving voice the existential threat of artificial intelligence. It first of all entails multiple positions linked to different groupings and statements in the operational formation.

As I will suggest, neural nets are particularly interesting because they re-draw human-machine boundaries through a combination of feeding-forward of potentials and propagating backwards of differences specifically concerned with images. Similarly, the practice of machine learning shifts subject positions in a backward and forward movement. It propagates potentializing optimism even as it undercuts the very differences that give rise to that optimism.

```
techniques year
```

1 insider threats 2013 2 anomaly detection 2013 3 naive bayes classification. 2013 4 social media 2013 5 facebook 2013 6 gender 2013 [1] 510216 3 discipline techniques year 1 statistics bayesian network 2013 2 statistics upper gastrointestinal 2013 3 statistics medical support 2013 4 statistics asymptotic analysi 2013 5 statistics binary discrimination 2013 6 statistics high dimensional 2013 [1] "neural network" "clustering"

[3] "k mean" "feature selection"

[5] "decision tree" "genetic algorithm"

[7] "ensexpectation maximizationble" "pattern recognition"

[9] "naive bayes" "random forest"

[11] "feature extraction" "association rule"

[13] "sexpectation maximizationi" "time serie"

[15] "maximum likelihood" "rough set"

[17] "algorithm" "knowledge discovery"

[19] "sexpectation maximizationantic" "nearest neighbor"

Almost every machine learning class, textbook, demonstration, and in recent years, machine learning competitions at some point turns to neural nets. Neural nets display, however, some instability in the research literature. Figure 8.1 shows the most frequent keywords for technical publications across the three main disciplinary domains

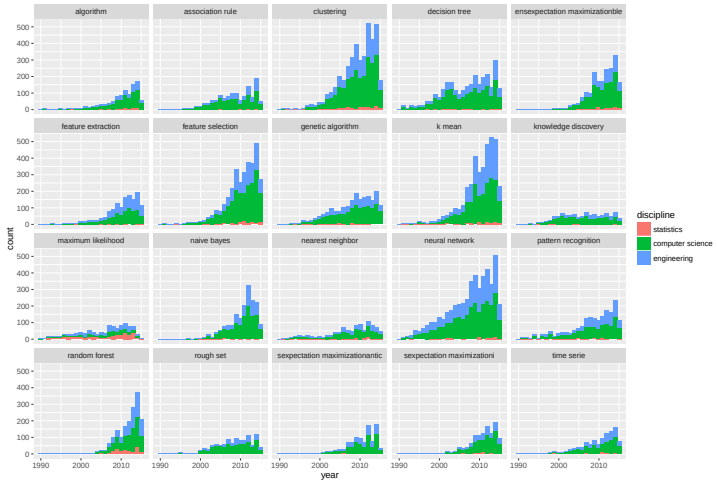


Figure 8.1: Techniques and concepts most frequently mentioned in machine learning publication keywords, 1955-2015

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 in the rankings. The prominence of neural nets on the engineering side of machine learning suggests a specific enunciative mapping.

Neural nets are often described from a deeply split perspective. At some moments, the description turns towards human subjects, or at least, the brains of human subjects. At other other moments, neural nets turn towards the vectorisation of data. Neural nets constantly oscillate between brain and information infrastructure. In some ways, they renew long-standing cybernetic hopes of bring brains and computation together in models of computational intelligence and agency (Hayles 1999). Although they stem from a biological inspiration (dating at least back to the work by 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 but hard to control. In

the course of fifty years, their serial re-invention — from perceptron via neural net to deep belief net — triply re-distributes subject positions amidst infrastructural re-configurations and vectorisation.

For instance, writing in the 1980s, David Ackley, Geoffrey Hinton (an important figure in the inception of neural nets over several decades), and Terrence Sejnowski link neuroscience and semiconductors:

> 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 (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 reproduce the plasticity of neuronal networks in the parallel distributed processing enabled by very densely packed semiconductor circuits. The problem here was how to organize these connections without hardwiring domain specificity into 'the architecture of the network.' How could the architectures adapt to the problem in hand?

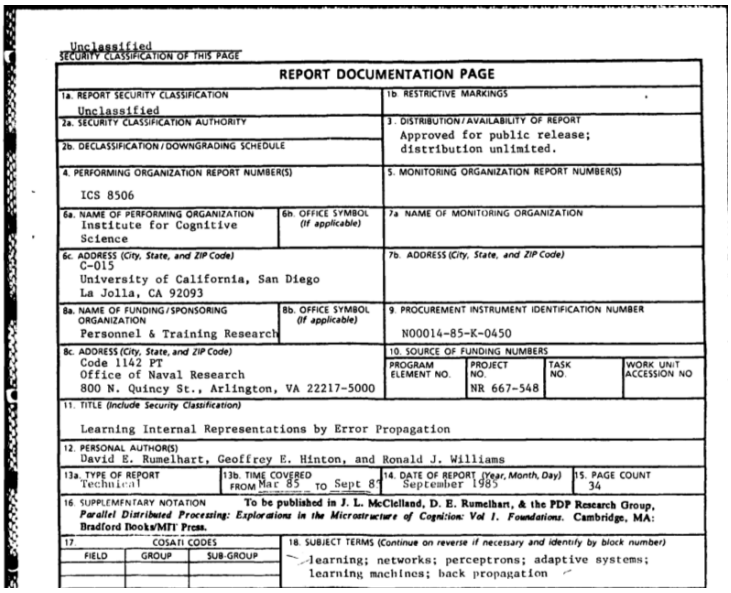
We saw in chapter 2 that the psychologist Frank Rosenblatt's perceptron (Rosenblatt 1958) first implemented McCulloch and Pitts' cybernetic vision of neurones as models of computation (Edwards 1996). While the computer science research on the perceptron wilted under criticism from artificial intelligence 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 multilayer perceptrons). In the mid-1980s, they developed the back-propagation algorithm (Rumelhart, Hinton, and Williams 1985; Hinton 1989), a way of adjusting the connections known as weights or parameters between nodes (neurones) in the network in response to features in the data (see Figure 8.2).

The back-propagation algorithm directly addressed the problem of learning to modify network organization without reliance on problem-dependent architectures, and in without having to program them in. Effectively, it constructs an architecture of generalization. While cognition, and the idea that machines would be cognitive (rather than say, mechanical, calculative, or rule-based) mesmerised 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 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

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



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. NR 667-548 / TASK NO. / WORK UNIT ACCESSION NO
11. TITLE (Include Security Classification) Learning Internal Representations by Error Propagation	
12. PERSONAL AUTHOR(S) 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, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Vol 1. Foundations.* 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 8.2: An early publication of the back-propagation algorithm (Rumelhart 1985)

chapter 7) 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 learning, and around the production of statements by machine learners, both human and non-human machine learners are assigned new positions. These positions are sometimes hierarchical and sometimes dispersed. The machine learner subject position 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 vectorize, optimize, probabilise, differentiate and refer, 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 vector space), and competing forms of accumulation or positivity. As Suchman suggests, examining privileged machines such as neural nets is a way to pay attention to the dispersed and somewhat disconnected sites from which subjects program, observe, design and respond to machine learners.

Competitive positioning

How do neural nets come to oscillate between different subject positions? The ranking of keywords in table ?? suggests that machine learning as an operational knowledge formation attributes a privileged and constitutive function to neural nets. Neural nets concurrently 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).

The forms of this field-making bridging are various.[5] The primary meeting point of different disciplines has perhaps been the machine learning competitions of the 1990s ~~that~~ pitted neural nets against other machine learners such as support vector machine. Many of these competitions focused on vision-related problems such as

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.

recognising handwritten numerals. 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 and during the 1990s, much effort focused on crafting neural nets to recognise these 60,000 or so handwritten digits.

Elements of Statistical Learning devotes a lengthy section to the analysis of image recognition competitions that began in the early 1990s and continue today. Like Alpaydin, it affirms 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).

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" (408–409). During the 1990s, zipcodes on envelopes (the set of handwritten digits we have already seen in chapter 6, the `mnist` ~~datasets (LeCun~~ and Cortes 2012)~~)~~ became a primary focus of learning. 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.

Although they brought statistics and computer science closer, neural networks have had a somewhat problematic position in machine learning. Even in relation to the paradigmatic handwritten digit recognition problem, neural nets struggled to gain purchase precisely because a human subject position remained intimately interwoven into their operation. 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' (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 point to a transformation in the work of machine learning:

> 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).

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 move~~s~~ 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*

The somewhat vacillating presence of neural nets in the machine learning literature itself finds parallels in the movements of individual machine learners. 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

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.

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 scale (for instance, in classifying videos on Youtube). In contemporary 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 handcrafted best-in-class 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 amidst the 'awesome materiality' of social media image streams 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 such as back-propagation used in neural nets 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

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).

often the case in machine learning, their proliferation occurs through re-distributions of knowledge and infrastructure associated with altered subject positions. While machine learners in their machine form can be assigned a privileged position in the transformations of knowledge and action, human machine learners are not exactly marginalized, at least in celebrated cases such as Ng, Le Cun, Hinton and others. Rather, we can see varying subject positions emerging in relation to specific devices and data forms (images, sounds, documents) in specific sites (social media platforms and mobiles devices in particular).

A varying subject position surfaces in the operational architecture of neural nets. Despite differences in diagrammatic form, 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 well-established transformations. '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:

$$
\begin{aligned}
Z_m &= \sigma(\alpha_0 m + \alpha_m^T X) m = 1, \ldots, M \\
T_k &= \beta_0 k + \beta_k^T Z, k = 1, \ldots, K, \\
f_k(X) &= g_k(T), k = 1, \ldots, K,
\end{aligned}
\tag{8.1}
$$

where $Z = (Z_1, Z_2, ..., Z_M)$, and $T = (T_1, T_2, ..., T_K)$. The activation function $\sigma(v)$ is usually chosen to be the sigmoid $\sigma(v) = 1/(1 + e - v)$

(392)

Equation 8.1 diagrams some familiar elements as well as some novelty. Some elements of the neural net are already familiar from linear models. The neural networks transform 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.

The apparently increasing power of neural nets to learn (to see, ~~to~~ find, ~~to~~ classify, ~~to~~ rank, ~~to~~ predict) owes much to diagrammatic substitutions 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 distinguish~~es~~ neural nets from the linear models ~~it~~ assemble~~s~~.

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j \tag{8.2}$$

Whereas the standard linear model shown in ~~E~~quation 8.2 indexes a single vector space X_j and approximates it using a single function $\hat{Y}$ by searching for the values of the parameters β_j that best incline a flat plane through the data, the three lines shown in equation 8.1 are woven through each other much more consecutively. Each line~~s~~ derives 'features' from the line above, adding layers to the network. So-called 'hidden layers,' such as line two of equation 8.1 repeatedly transform the vector space inside the model ~~itself~~. Each node, 1..K,

adds a new dimension to this internal vector space. In many layered deep learning neural nets, the dimensionality of the vector space vastly expands. Much hinges on the unobstrusive 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' (Hastie, Tibshirani, and Friedman 2009, 394). σ, it seems, allows neural nets to generalize beyond the linear model.[9]

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.

Varying subject positions in code

The operational diagram of neural nets, I would suggest, ascribes subject positions associated with it. How does that happen? Some familiar diagrammatic operations immediately appear in almost any actual example of a neural net. In the code vignette shown below, the dataset 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`.

Listing 8.1: Neural net for `titanic` dataset

````
```r
library(neuralnet)
titanic = read.csv("data/titanic3.csv", stringsAsFactors =
    ↪ FALSE)
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 + parch +
    embarkedC + embarkedQ + embarkedS, data = titanic_train,
        ↪ err.fct = "ce",
    linear.output = FALSE, rep = 1, hidden = 3, stepmax = 1e
        ↪ +05)
titanic_test = titanic_transformed[-train_index, ]
test_error = round(sum(0.5 < compute(titanic_net, titanic_test
    ↪ [, -c(1, 2)])$net.result)/sum(titanic_test$survived),
    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` library (Fritsch and Guenther 2012) closely resembles the lines of code used to construct linear models for the `prostate` data (see chapter 3). 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. 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 machine learners turn to them?

Perhaps the only distinctive feature of the code listing 8.1 appears in the expression `hidden = 3`. This architectural feature 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. These 'hidden units' are key to neural net since they construct the 'derived features' that the model learns from the input data X.

How many nodes are hidden and in what topology matters less than the existence of operation that allows their topology to be configured in an encounter with data. The novelty of this operation was central to research into neural nets. 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' 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 Rumelhart 1986)~~)~~.

The hidden layers lend a network form to machine learning. The final diagrammatic form in which neural nets appear is the network. 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 ~~F~~igure 8.3.

Although the network topology of the model appears in many more complicated forms, it diagrams several operations. 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, suggests receptive or recording surface (~~for instance~~, a camera). Early neural net papers on the handwritten digital recognition problem sometimes describe cameras mounted above tables focused on images (LeCun et al. 1989). Second, it presents an output layer that can contain single or multiple nodes, the k of equation 8.1. In

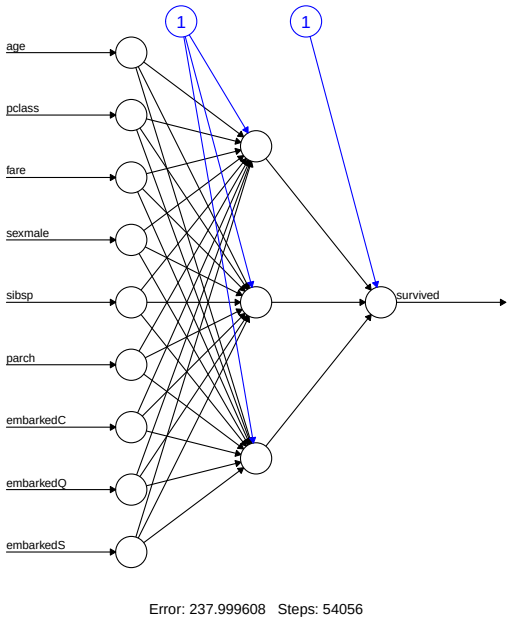


Figure 8.3: Neural network topology for 3-hidden unit 'titanic' data

the `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 exhibits ordered forms of movement. Data and calculation propagate from left to right or ~~vice-versa~~. (Sometimes the networks are rotated~~,~~ and the flow is vertical~~,~~ 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 collapse into 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 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.

The subjects of a hidden operation

How do 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, assign subject positions? How would we describe the figure of human machine learner in this setting? Is the human machine learner like Vlad, the former Eastern European mathematician tending the training of a neural net at the heart of the call centre knowledge management system, or more like Heather Arthur, the programmer who wrote `kittydar`?

When in *The Archaeology of Knowledge*, Foucault presents the 'position of the subject' as an anchor point for the power-laden, knowledge forming enunciative functions, he does not identify it as 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' (Foucault 1972, 54). Positions derive from operations that determine statements that become a kind of law for the subject.

As Foucault puts it, in a mathematical example and a conceptual formulation that broadly anticipates accounts of performativity,

> 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).

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 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, therefore, 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 defines 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.

Algorithms that propagate errors

The distinctive feature of neural nets, at least in their ordinary 'vanilla' forms, consists in their use of a series or chain of gradient descent operations to minimise errors by adjusting the weights (or parameters) of all the nodes (or linear models) comprising the network. 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 '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. Does an algorithm such as back-propagation diagram the slippery coincidence of subject of operation and subject of statement in machine learning?

The subject-positioning zone of slippage between statement and operation appears as error. 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, sites 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 must have more ore less disappeared as automatic mail sorting improved).) 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 gripped or seen by a knowing or controlling subject, since on this account, the operations and statements in their dispersions, accumulations and distributions overflow any simple dyadic relation between a subject-object or human-machine/world.

As we have seen in chapter 4, error rates, training error, test error, generalization error, validation error are just some of the errors that criss-cross between human and machine learners. Errors render operations as 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 statements of errors. Every machine learner makes different determinations in relation to model parameters and errors. We have already seen something of the forward movement that runs between the input layer and the output layer with its classificatory statements. Equations 8.1 imply data moving a succession of layers and their nodes. Conversely, the back-propagating phase of a neural net move from output towards input layer updating weights of various nodes in the light of differences between predicted and known outputs.

$$\beta_{km}^{(r+1)} = \beta_{km}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \beta_{km}^{(r)}}$$
$$\alpha_{ml}^{(r+1)} = \alpha_{km}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \alpha_{km}^{(r)}} \tag{8.3}$$

(Hastie, Tibshirani, and Friedman 2009, 396)

Many different parameters figure in the back-propagation functions shown in 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 ($\sum$) and partial differentiation (∂). The usual indexical relations to vector space 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. Partial derivatives express the sensitivity of errors with respect to the weights or parameters of the nodes. 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 update as the model is trained on the input data. The two lines of equation 8.3 specify how first the values of the parameters β_{km} 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 weights α_{km} 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 8.3.

'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]

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.

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 hand-written 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

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 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/kdd-dcup/index.php](KDD 2013) 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). Some competitions have recruitment opportunities 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).

While most employment agencies rely on CVs (curriculum vitae), Kaggle operates more like feed-forward and back-propagation between multiple competitions as a way of optimising its ranking of machine learners.

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 —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. As we have seen frequently, accumulation, aggregation and 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).

11. At the time of writing, Kaggle claims around 320,000 competitors.

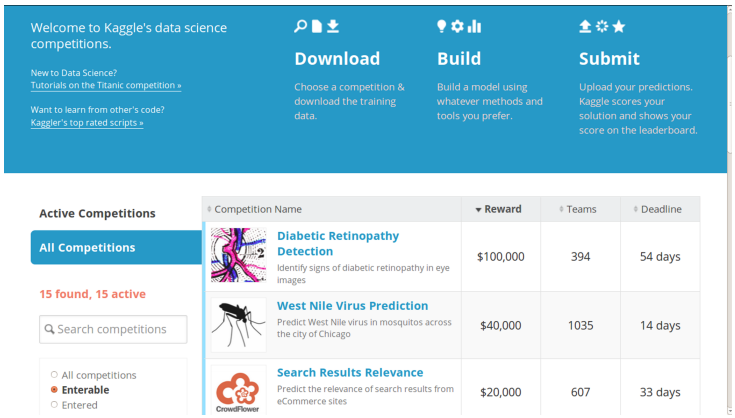


Figure 8.4: Kaggle data science competitions

Like a neural network with many layers and nodes, competitions subject competitors (several hundred thousand in Kaggle) 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 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 leaderboard 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 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.[12] 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.

12. 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 optimise the fit of a subject position to an operational formation.

Superimposing power and knowledge

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 take the form of examinations that set a problem, define some limits or constraints on its 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 instance 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 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 links statements and operations. Examinations combine ceremony, ritual, experiment, force and truth in subject and object positioning operations. The consolidation of machine learning as a data practice today in competitions occurs via a much more pervasive practice of examining and testing. The forms of visibility created by competitions individualize and normalize machine learners (often by proper names), and optimise extractions of force, time, propensities and aptitudes.

In many Kaggle competitions (some titles are shown in table 8.1), winning entries come from machine learners working together. In the National Data Science Bowl competition of 2015, competitors were asked to classify images of more than 100 species of plankton. 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 elements 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

Competition	Reward_amount	domain	data_type
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 develop safe and effective medicines by predicting molecular activity	40000	medicine	measurements
West Nile Virus Prediction Predict West Nile virus in mosquitos across the city of Chicago	40000	science	measurements
Acquire Valued Shoppers Challenge Predict which shoppers will become repeat buyers	30000	retail	transaction
Driver Telematics Analysis Use telematic data to identify a driver signature	30000	traffic	measurements
Restaurant Revenue Prediction Predict annual restaurant sales based on objective measurements	30000	retail	attributes
Caterpillar Tube Pricing Model quoted prices for industrial tube assemblies	30000	retail	attributes
GigaOM WordPress Challenge Splunk Innovation Prospect Predict which blog posts someone will like	25000	social_media	texts
U S Census Return Rate Challenge Predict census mail return rates	25000	government	actions
Belkin Energy Disaggregation Competition Disaggregate household energy consumption into individual appliances	25000	energy	actions

Table 8.1: The highest prize money machine learning competitions on Kaggle

(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 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 plain or 'vanilla' neural nets discussed above, 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 a range of learners into the operation, just as Kaggle itself networks many machine learners through its competitions.

A massive parallel computation allows 'deep' learning. Infrastructure and cognition entwine heavily here, since the very possibility of training large 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

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

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).

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.'

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 at the conjunction of operation and the statements in machine learning. Machine learners such as deep belief nets adjust and align subject positions through their many convolutional layers. They supplant, for instance, the skilled configuration of feature engineering that characterised work on decision trees, linear regressions, support vector machines and predecessor neural nets (and appears as a key element in figure 8.1). Similarly, they absorb the professional skills of Go players in training models that win against the best human players (Silver et al. 2016). The subject position of a machine learner occupies a zone of diagrammatic slippage between