7. Optimising machine learners to learn on their own

to do

- possibly add the convolutional aspect of this – feature learning – isn't this an important shift
  - convolution could go along with back-propagation and recurrence as ways in which statements and operations are superimposed
- pick up on the N=ALL argument – now that joint probability can be modelled, then many more generative models
- the programming of ml as the backpropagation and feed forward of machine learning as function finding into finding a function for machine learning
- deep neural nets (Hassabis et al. 2013) should appear here
- the Higgs Boson challenge
- MF in d&p writes about how the individual internalises the disciplinary mechanism – we are all machine learners now?
- the masculinity of machine learning – how to deal with that? some prominent women, but massively masculinist – takes me back to 1996 publication - also use SI of angelaki on geophilosophy of masculinity. See zotero masculinity folder
- the people change alongside the data; their sense of the power of data has a cost for them too
- put in Perlich stuff about data leakage – really important to focus on competition as a way of showing how people do things

key points for framing

- error is the cross-over
- examination is what allows both parallel architectures and competitions to come together
- examination makes each individual a case – not just the data, but the machine learners are examined
- my own sense of uncertainty about this stuff, and mixtures of optimism and sense of having to keep moving forward, of needing to re-calibrate, to go in parallel, to avoid overfitting myself, but at the same complexifying what I'm doing.

Introduction

> 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 2015a)

Here are the words of Hilary Mason, Chief Scientist at bitly.com (an online service that shortens URLs), at a London conference in 2012 called 'Bacon: Things Developers Love':

> You have all of these things that are different – engineering, infrastructure, mathematics and computer science, curiosity and an understanding of human behaviour – that is something that usually falls under the social sciences. At the intersection of all these things are wonderful people. But we're starting to develop something new, and that is - not that all of these things have not been done for a very long time - but we are only just now building systems where people, individual people, have all of these tools in one package, in one mind. And there are lots of reasons this is happening now. But its a pretty exciting time to be in any of these things. (Hilary Mason, Chief Scientist, bitly.com) (Bacon 2012)

These propositions, if they amount to a *statement* in Michel Foucault's sense of that term, assign subject positions . In what ways? In front of an audience of several hundred software developers, Mason describes shifts in the work of programming associated with the growth of large amounts of data associated with 'human behaviour.' At the centre of this shift stand 'wonderful people' who combine practices and knowledges of communication infrastructure, technology, statistics, and 'human behaviour' through curiosity and technical skills. Mason was, in effect, telling her audience of software developers who they should become and at the same time pointing to the some of the expansive changes occurring around them. The title of her talk was 'machine learning for hackers', and her audience were those hackers or programmers whose attention may have been previously trained on web interfaces or database queries, but was now drawn towards machine learning. A change in programming practice and a shift towards machine learning was, she implied, the key to programmers becoming the wonderful people, agents of their own time, capable of doing what is only now just possible because it is all together in 'one package, one mind.' Note that

concatenation of 'one package, one mind' does not definitively allocate agency to people or things. Here, I would tentatively suggest, Mason adumbrates the outline of an agent of anticipation, someone or something at the intersection of network infrastructure, mathematics and human behaviour.[1] Mason, one of *Fortune* magazines 'Top 40 under 40' business leaders to watch (CNN 2011) and also featured in *Glamour*, a teenage fashion magazine (Mason 2012), might personify such a wonderful person. She is not a lone example.[^8.2] Equally so, she might *a-personify* the intersection.

'It is the privileged machine in this context that creates its marginalized human others' writes Lucy Suchman in her account of the encounters that 'effect "persons" and "machines" as distinct entities' (Suchman 2006, 269) . While Mason and other relatively well-known human machine learners are not exactly marginalized (just the opposite, they achieve minor celebrity status in some cases), Suchman recommends 'recognition of the particularities of bodies and artifacts, of the cultural-historical practices through which human-machine differences are (re-)iteratively drawn, and of the possibilities for and politics of redistribution across the human machine boundary' (285). The intersections that machine learners currently occupy are heavily re-distributional. In almost every instance, machine learners claim to do something that humans alone, no matter how expert, could not. (We need only think of the demonstration of the radio-controlled helicopter flying upside down that Andrew Ng shows in his introduction of machine learning lecture ("Lecture 1 | Machine Learning (Stanford). Youtube" 2008) .) Could the 'wonderful people' that Mason describes

[1] In earlier work on machine learning (Mackenzie 2013), 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 Vincanne Adams, Michelle Murphy and Adele Clarke (**???**). 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, the focus is less on transformations in programming practice and software development, and more on the asymmetries of different machine learner subjects.

also be seen as marginalized human others? Does the re-distribution of engineering, mathematics, curiosity, infrastructure and 'something that usually falls under the social sciences' (but perhaps no longer does so?) both energise subjects ('its a pretty exciting time to be in any of these things') and assign them a marginal albeit still pivotal position?

The potentials of re-distribution are the topic of this chapter. The key machine-diagram I draw on here is artificial neural networks, or neural nets, in their various forms ranging from the perceptron, through the multilayer perceptron (MLP), the convolutional neural nets (CNN), recurrent neural nets (RNN) and deep belief networks or deep neural networks of many recent deep learning projects (particularly in machine learning competitions, as discussed below (Dahl 2013)). For present purposes, neural nets embody a propagating intersection between infrastructures, engineering and human behaviour (as Mason puts it).Neural networks re-draw human-machine differences in specific ways. A straightforward example of this appear in neural net publications. Geoffrey Hinton, Simon Osindero and Yee-Why Teh writing in *Neural Computation* in 2006 described a 'fast learning algorithm for deep belief nets' (Hinton, Osindero, and Teh 2006). Their description, whilst mostly couched in terms of conditional distributions, parameters, and generative models also contains a section entitled 'Looking into the Mind of a Neural Network' (1545-1546). In that section, they describe how they used their deep belief network to generate rather than classify images.[2] In the process they were able to see what the 'associative memory has in mind' (1545). The term 'mind' it turns out 'is not intended to be metaphorical' (1546) because the neural net in question has a distributed memory of the digits it has seen. Put slightly more formally, 'the network has a full generative model, so it is easy to look into its mind -

[2]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.

we simply generate an image from its high-level representations' (1529). The conjunction or intersection of mind and generative model is in many respects a new element of artificial intelligence (which has typically relied on rule-based or symbolic reasoning), but I think that the way in which 'mind' appears here in the form of a generative model (see chapter **??**) covers over other layers of positioning and subjectification that might be more interesting to explore in their specificity (for instance, in the work done on specific artifacts such as images) than in terms of the general existential threat of artificial intelligence. Neural nets re-iterate 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 assigns subject positions in a backward and forward movement that propagates potentialising optimism even as it undercuts the very differences that give rise to that optimism.

The recurrence of neural nets in machine learning

In almost every machine learning class, textbook, demonstration, and in recent years, in machine learning competitions, neural nets make some appearance. They have an ambivalent status. Neural nets are often described from a deeply split perspective that in some points turns towards human subjects, or at least, the brains of human subjects, and in other ways towards the ongoing expansion of computational platforms. In some ways, they renew long-standing cybernetic hopes of brains and cognition as models of computational power. Although they stem from a biological inspiration (dating at least back to the work on McCulloch and Pitts in the 1940s (Halpern 2015; Wilson 2010)), they gain traction first in the 1980s and then again from mid-2000s onwards, as ways of dealing with changing computational infrastructures, and the difficulties of

capitalising on infrastructure that is powerful yet difficult to manage. This triple re-invention – from perceptron via neural net to deep belief net – is heavily re-distributional. There is an almost constant oscillation between brain and information infrastructure running now for almost a half a century in neural nets. For instance, David Ackley, Geoffrey Hinton (an important figure in the inception of neural nets during the 1980s and in the revival of neural net in the form of deep learning in the last decade), and Terrence Sejnowski wrote in the early 1980s:

> Evidence about the architecture of the brain and the potential of the new VLSI technology have led to a resurgence of interest in "connectionist" systems … that store their long-term knowledge as the strengths of the connections between simple neuron-like processing elements. These networks are clearly suited to tasks like vision that can be performed efficiently in parallel networks which have physical connections in just the places where processes need to communicate. … The more difficult problem is to discover parallel organizations that do not require so much problem-dependent information to be built into the architecture of the network. Ideally, such a system would adapt a given structure of processors and communication paths to whatever problem it was faced with (Ackley, Hinton, and Sejnowski 1985, 147-148).

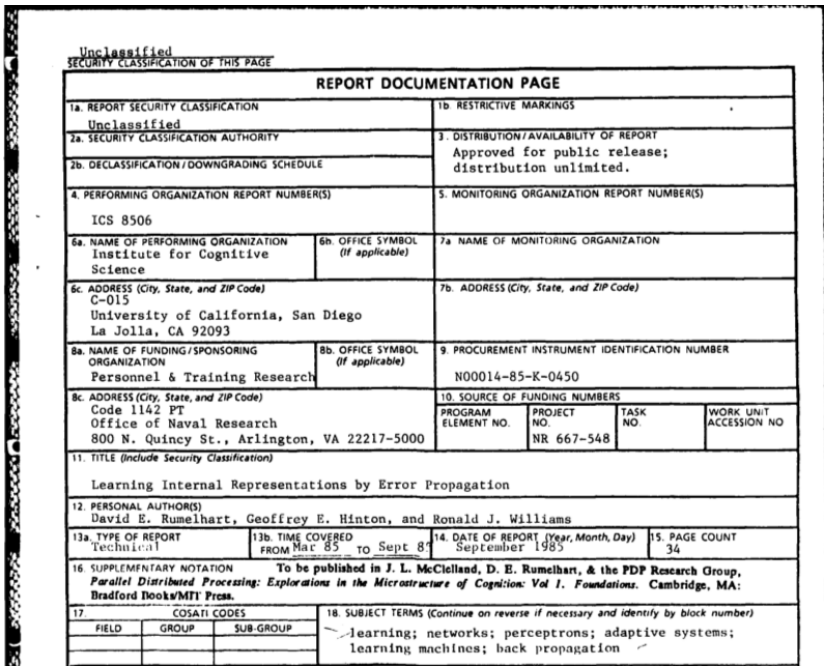
This convergence between brain and 'new VSLI [Very Large Scale Integrated] technology' – semiconductor chips – sought to map the plasticity of neuronal networks onto the parallel distributed processing enabled by very densely packed semiconductor circuits. The problem here was how to organize these connections without having to hardwire domain specificity into 'the architecture of the

network.' How could the architectures adapt to the problem in hand?

We saw in **??** that the psychologist Frank Rosenblatt's perceptron (Rosenblatt 1958) first implemented McCulloch and Pitts' cybernetic vision of neurones as models of computation (Edwards 1996) . While the perceptron did not weather the criticism of artificial intelligence experts such as Marvin Minsky (Minsky famously showed that a perceptron cannot learn the logical exclusive OR or `XOR` function; (Minsky and Papert 1969)), cognitive psychologists such as David Rumelhart, Geoffrey Hinton and Ronald Williams returned to work 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 between nodes (neurones) in the network in response to features in the data (see Figure 1). The back-propagation algorithm directly addressed the problem of constructing parallel network organizations without reliance on problem-dependent architectures. Effectively, an architecture of generalization was implemented. While cognition, and the idea that machines would be cognitive (rather than say, mechanical, calculative, or even algorithmic) constantly organised research work in artificial intelligence for several decades, the development of the back-propagation algorithm as a way for a set of connected computational nodes to learn also had strong infrastructural resonances. These resonances became much more visible from around 2006 when 'deep belief nets' appeared as a way of training many-layered neural nets (Hinton and Salakhutdinov 2006a). These resonances continue to echo today and indeed attract much attention.[^7.005] Like the advent of VSLI in the early 1980s, the vast concentrations of processing units in contemporary data centres (hundreds of thousands of cores as we saw in the case of Google Compute in the previous chapter **??**) and even in the graph-

ics cards developed for high-end gaming and video rendering (GPUs for PC gaming now typically have a thousand and sometimes several thousand cores) pose the problem of organizing infrastructure so that processes can communicate with each other. Machine learners may well become more important as loose or mutable infrastructural arrangements than as epistemic instruments. [^7.005]: 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*(**???**), or *The Guardian* (**???**). In many of these accounts, machine learning and neural nets in particular appear both in the guise of the existential threat of artificial intelligence and as a mundane device (for instance, speech recognition on a mobile phone). The spectacular character of deep learning could be analysed in terms like that of the genomes discussed in chapter **??**. In both cases, the advent and transformation of these machine learners is closely linked to networked platforms (such as Google, Facebook, Yahoo and Microsoft) and their efforts to encompass within their services as many elements of experience, exchange, communication and power as possible. Deep learning machine learners currently focus mostly on images (photographs and video) and sounds (speech and music), and usually attempt to locate and label objects, words or genres.

The back-propagation or 'backprop' algorithm will return below, but for the moment, this oscillation between cognition and infrastructures, between people and machines, itself suggests another way of thinking about how 'long-term knowledge' takes shape today. At the same time as infrastructural reorganization takes place around learning, and around the production of statements by machine learners, both human and non-human machine learners are assigned



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 1: An early publication of the back-propagation algorithm: Rumelhart, Hinton and William's 1985 paper [@Rumelhart_1985]

new positions. These positions are dispersed, hierarchical and distributed. The subject position in machine learning is a highly relational one rather than a single concentrated form of expertise (as we might find in a clinical oncologist, biostatistician or geologist). It is intimately bound and connected to transforms in infrastructure, variations in referentiality (such as we have seen in the construction of the common vector space), and competing forms of authority (as we have seen in the accumulations of different techniques). As Suchman suggests, examining privileged machines is a way to pay attention to variously marginalized human others.

Some machine learners attribute a more privileged and constitutive function to neural nets. Neural nets synchronically spread into many difference disciplines: cognitive science, computer science, linguistics, adaptive control engineering, psychology, finance, operations research, etc., and particularly statistics and computer science during the 1980-1990s. This dendritic growth did not just popularise machine learning. It brought engineering and statistics together more strongly. As Ethem Alpaydin 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. We saw some use of neural nets in genomics in the previous chapter (**??**). But the primary space of coexistence of different disciplines around machine learning has perhaps been competitions during the 1990s that pitted neural nets against other machine learners in classifying handwritten numerals such as zipcodes on envelopes, and in particular one set of handwritten digits that remain in constant use, the

MNIST (Modified National Institute of Standards) dataset (LeCun and Cortes 2012). The many competitions focused on the MNIST dataset are, I suggest, a form of demonstration and testing of machines and people that matter greatly to re-iterative drawing of machine-human differences in machine learning. It is no accident that Hastie and co-authors in *Elements of Statistical Learning* devote a lengthy section to the analysis of these handwritten digit recognition competitions that began in the early 1990s and still today. Like Alpaydin, they affirm the coordinating effect of these competitions on the development of machine learning:

> This problem captured the attention of the machine learning and neural network community for many years, and has remained a benchmark problem in the field (Hastie, Tibshirani, and Friedman 2009, 404).

The handwritten digits used in these competitions, particularly the Neural Information Processing System workshops and KDD Cup (Knowledge Discovery and Data Mining) (KDD 2013), all come from the MNIST dataset and during the 1990s, much effort focused on crafting neural nets to recognise these 60,000 or so handwritten digits. As Hastie and co-authors observe, 'at this point the digit recognition datasets became test beds for every new learning procedure, and researchers worked hard to drive down the error rates' (Hastie, Tibshirani, and Friedman 2009, 408-409).

Despite this binding function, neural networks have had a somewhat problematic position in to machine learning. Even in relation to the paradigmatic handwritten digit recognition problem, neural nets struggled to gain purchase. On the one hand, their analogies and figurations as sophisticated neuronal-style models suggested cognitive capacities surpassing the more geometrical, alge-

braic 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. Somewhat loopily, the competition to automate recognition of handwritten digits entailed much handcrafting and recognition of variations in performances of the digital.

Neural nets also receive uneven attention in the machine learning literature. In Andrew Ng's Stanford CS229 lectures from 2007, they receive somewhat short shrift: around 30 minutes of discussion in Lecture 6, in between Naive Bayes classifiers and several weeks of lectures on support vector machines ("Lecture 6 | Machine Learning (Stanford)" 2008). As he introduces a video of an autonomous vehicle steered by a neural net after a 20 minute training session with a human driver, Ng comments that 'neural nets were the best for many years.' The lectures quickly moves on to the successor, support vector machines. In *Elements of Statistical Learning*, a whole chapter appears on the topic, but prefaced by a discussion of the antecedent statistical method of 'projection pursuit regression.' The inception of 'projection pursuit' is dated to 1974, and thus precedes the 1980s work on neural nets that was to receive so much attention. In *An Introduction to Statistical Learning with Applications in R*, a book whose authors include Hastie and Tibshirani, neural nets are not discussed and indeed not mentioned (James et al. 2013). Textbooks written by computer scientists such as

Ethem Alpaydin's *Introduction to Machine Learning* do usually include at least a chapter on them, sometimes under different titles such as 'multi-layer perceptrons' (Alpaydin 2010). Willi Richert and Luis Pedro Coelho's *Building Machine Learning Systems with Python* likewise does not mention them (Richert and Coelho 2013). Cathy O'Neil and Rachel Schutt's *Doing Data Science* mentions them but does not discuss them (Schutt and O'Neil 2013), whereas both Brett Lantz's *Machine Learning with R* (Lantz 2013) and Matthew Kirk's *Thoughtful Machine Learning* (Kirk 2014) devote chapters to them. In the broader cannon of machine learning texts, the computer scientist Christopher Bishop's heavily cited books on pattern recognition dwell extensively on neural nets (Bishop and others 1995; Bishop and Nasrabadi 2006). Amongst statisticians, Brian Ripley's *Pattern Recognition and Neural Networks* (Ripley 1996), also highly cited, placed a great deal of emphasis on them. But these specific documents against a pointillistic background of hundreds of thousands of scientific publications mentioning or making use of neural nets since the late 1980s in the usual litany of fields – atmospheric sciences, biosensors, botany, power systems, water resource management, internal medicine, etc. This swollen publication tide attests to some kind of formation or configuration of knowledge invested in these particular techniques, perhaps more so than other I have discussed so far (logistic regression, support vector machine, decision trees, random forests, linear discriminant analysis, etc.).

The shifting fortunes of neural nets are frequently discussed in contrasting terms by machine learners themselves, but in recent years they share an awareness of some kind of transformation:

> Neural networks went out of fashion for a while in the 90s - 2005 because they are hard to train and other techniques like SVMs beat them on some problems. Now people have figured out better meth-

> ods for training deep neural networks, requiring far fewer problem-specific tweaks. You can use the same pretraining whether you want a neural network to identify whose handwriting it is or if you want to decipher the handwriting, and the same pretraining methods work on very different problems. Neural networks are back in fashion and have been outperforming other methods, and not just in contests (Zare 2012).

The somewhat vacillating presence of neural nets in the machine learning literature itself finds parallels in the fortunes of individual machine learners. 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` led many academic 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 2006b) delivers him to Google in 2013. These trajectories between academic research and industry are not unusual. 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.[3] Ng

[3]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

himself in 2014 began work as chief scientist for the Chinese search engine, Baidu leading a team of AI researchers specializing in 'deep learning,' the contemporary incarnation of neural nets (Hof 2014) . In recent years, (2012-2015), work on neural nets has again intensified, most prominently in association with social media platforms, but also in the increasingly common speech and face recognition systems found in everyday services and devices. Many of these neural nets are like `kittydar` , but implemented on a much larger and more distributed 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 best performer in the machine learning image classification competitions of the 1990s, from second best competitor in late 1990s to the spectacular promise of deep belief networks in 2012, suggests that some powerful dynamics or becomings are in play around them. These dynamics are not easily understood in terms of celebrity machine learners (human and non-human) suddenly rising to prominent or privileged positions in the research departments of social media platforms.[4] Nor does it make sense to attribute the rising fortunes of the neural net to the algorithms themselves, as if some decisive advance occurred in algorithms. The algorithms used in neural net have not, as we will

was unsupervised (Markoff 2012). That is, the neural nets were not trained using labelled images of cats.

[4]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).

see, been radically transformed in their core operations since the 1980s, and even then, the algorithms themselves (principally gradient descent) were not new. There have been important changes in scale (similar to those described in the previous chapter in the case of the `RF-ACE` algorithm and Google Compute), but as is often the case in machine learning, their re-invention occurs through proliferation, changes in scale, re-distributions of knowledge and infrastructure and specific optimisations. While machine learners in their machine form can be assigned a privileged position in the transformations of knowledge and action today, human machine learners are not exactly marginalized, at least in high profile cases such as Ng, Le Cun, Hinton and others. Rather, the scale of machine learning seems to be changing both for the algorithms and for the human computer scientists, programmers and engineers.

What accounts for this acceleration and slowing-down, the intensified interactions and abandonments of neural nets over the last three decades? Despite their apparent differences in origin, neural net share much with other machine learners. The language of brain, neurones and cognition associated with neural net covers over their much more familiar vector-space, function-finding and optimisations they rely on in practice. '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:

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

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 $\sigma(v) = 1/(1 + ev)$

(Hastie, Tibshirani, and Friedman 2009, 392)

Equation 1 is a diagram with some familiar elements as well as some novelty. Some of the diagrammatic operation of the neural net is already familiar from the linear models. The neural networks traverse data in a vector space denoted by X. That is common to nearly all machine learners. They make use of the non-linear sigmoid function that lies at the heart of one of the main linear classifiers used in machine learning, logistic regression . Their training and learning processes have come to rely on the same kinds of cost, loss or error functions we have seen in other machine learners. Their apparently increasingly power to learn (to see, to find, to predict) again seems to owe much to re-configuration, to the diagrammatic movements that recombine operations in new intersections.

There are, however, some differences in this diagram. Equation 1 has three lines rather than one, and this layering and its diagonal patterns of indexical referencing running between subscripts distinguishes neural nets from linear models more generally.

$$
\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j \tag{2}
$$

Whereas the standard linear model shown in Equation 2 indexes a single common vector space X_j and approximates a single function $\hat{Y}$ by searching for the values of the parameters β_j that best incline a plane through the given data, it seems that the three lines of the neural net model show in equation 1 are woven through each other much more consecutively than the linear regression and logistic regression models. 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.[5]

The common diagrammatic operations of neural nets and other supervised machine learners immediately appears in almost any actual example of a neural net. In the code vignette shown below, the data is a spreadsheet of information about passengers of the Titanic. The `titanic` dataset, like `iris` or `boston` is often used in contemporary machine learning pedagogy. It is for instance, the main training dataset used by (kaggle.com)[http://kaggle.com], an online machine learning competition site I will discuss below . The first few lines of the `R` code load the dataset and transform it into vector space. For instance, variables such as `sex` that take values such as `male` and `female` become vectors of `1` and `0` in a new variable `sexmale`.

```
library(neuralnet)
```

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

```
titanic = read.csv('data/titanic3.csv')
```

[5]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 [TBA - the Gorlot paper]

```
titanic_transformed = as.data.frame(model.matrix(~survived + age+ pclass + fare+ sibsp +
+ parch + embarked, titanic))
head(titanic_transformed)
```

```
##   (Intercept) survived   age pclass    fare sibsp sexmale parch embarkedC
## 1           1        1 29.00      1 211.3375     0       0     0         0
## 2           1        1  0.92      1 151.5500     1       1     2         0
## 3           1        0  2.00      1 151.5500     1       0     2         0
## 4           1        0 30.00      1 151.5500     1       1     2         0
## 5           1        0 25.00      1 151.5500     1       0     2         0
## 6           1        1 48.00      1  26.5500     0       1     0         0
##   embarkedQ embarkedS
## 1         0         1
## 2         0         1
## 3         0         1
## 4         0         1
## 5         0         1
## 6         0         1
```

```
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, hidden=5)
```

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

```
titanic_test = titanic_transformed[-train_index,]
test_error = round(sum( 0.5 < compute(titanic_net, titanic_test[,-c(1,2)])$net.result)/s
```

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

The line of the code that constructs a neural net using the `neuralnet` library (Fritsch and Guenther 2012), and the description of the classifier here is a familiar one. Despite its biological inspiration, the `R` formula for the neural net looks very similar to other machine learners. 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
```

As is normally the case in `R` model formula, the response or target variable `survived` is expressed as a combination of other variables. In this case, the plus sign `+` indicates that the combination is linear or additive. As Hastie puts, 'the central idea is to extract linear combinations of the inputs' or predictor variables. If this model formula looks so similar to other machine learning techniques we have been discussing, what do neural networks add? Why did and do so many people turn to them?

In the code vignette above, the expression `hidden = 5` points to the distinctive architecture of these models, an architecture that does not appear in the model formula in the `R` code but does figure in the lines of Equation 1. The hidden units are indexed in the first line of the model in the variables Z_m. These 'hidden units' are key to neural net since they construct the 'derived features' that the model learns from the input data X. As Rumelhart, Winton and Williams announce the algorithm in a letter to *Nature* in 1986 entitled 'Learning representations by back-propagating errors' :

> We describe a new learning procedure, back-propagation, for net-

> works of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure (Rumelhart, Hinton, and Williams 1986, 533).

Again, despite the persistent reference to biology, the description of the 'new learning procedure' starts to sound more like machine learning. There is talk of minimizing a measure of difference between actual and desire output vectors (optimizing through a cost or loss function), as well as mention of 'features' and 'weights' (usually a synonym for model parameters: 'the neural network model has unknown parameters, often called weights, and we seek values for them that make the model fit the training data well' (Hastie, Tibshirani, and Friedman 2009, 395)). The novelty, however, consists in the 'hidden' units whose interactions 'represent important features.' In other words, the flat additive combination of features expressed in the `R` model formula above does not convey the interactions of these units. As Hastie and co-authors put it, 'the units in the middle of the network, computing the derived features Z_m, are called hidden units because the values Z_m are not directly observed' (Hastie, Tibshirani, and Friedman 2009, 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 final major form in which neural net appear is the network diagram. Network graphs already appeared in Rosenblatt's perceptron work (Rosenblatt 1958), but they ramify tremendously in the aftermath of back-propagation. Almost every book and article relating to neural net presents some version of the diagram shown in Figure 2.

```
## Error in plot.nn(titanic_net, fontsize = 8, show.weights = FALSE): object 'titanic_net' n
```

```
## png
##   2
```

The network topology of the model appears in countless more complicated forms, and it practically seems to do several things in neural net literature. First, it presents a layer – the input layer – that indexes something in the world. The input layer, shown as X in the algebraic diagram, becomes like an organ, an eye or perhaps a camera. Early neural net papers on the handwritten digital recognition problem sometimes describe camera's mounted above tables (LeCun et al. 1989). Importantly, it presents an output layer that can contain single or multiple notes. In the `titanic` examples, a single target node (figures as T in the equations 1). In the `MNIST` handwritten digit recognition models, there are usually ten output nodes, one for each of the digits 0 … 9. Second, the network diagram presents some ordered forms of movement. Data and calculation propagate from bottom to top or vice-versa. (Sometimes the networks are rotated, and the flow is horizontal, but still bi-directional). Bi-directional hierarchical movement is key to the back-propagation algorithm in feed-forward and more complicated recurrent and convolutional neural nets. Third, it renders visible in

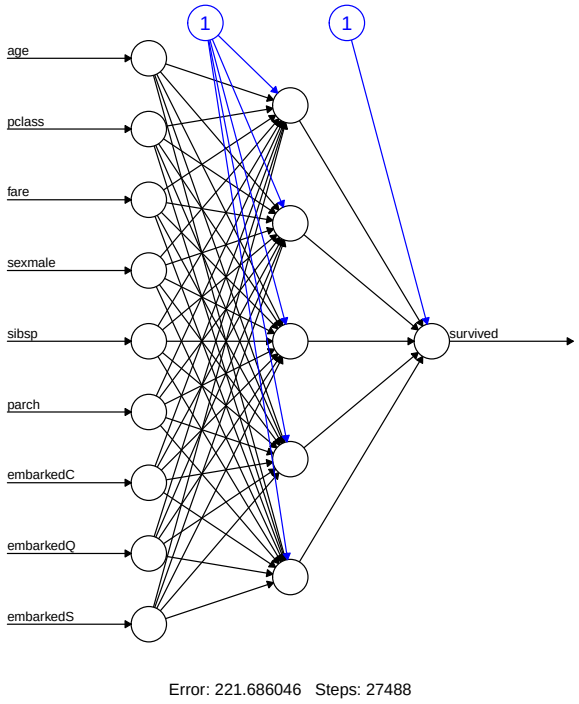


Figure 2: Neural network topology for ‘titanic‘ data

principle the vital hidden nodes. Without the hidden nodes, neural nets revert to linear models. With the hidden nodes, the Z_m of the equations 1, neural nets, like some other machine learners we have discussed such as support vector machines, effectively expand the common vector space by constructing new dimensions in it. The derived features or 'learned representations' (to use the language of (Rumelhart, Hinton, and Williams 1986)) can expand indefinitely, according to different network topologies. Hidden nodes and hidden layers can multiply, bringing many new interactions and relations into the model (as we see in more recent revivals of neural net as deep learning (Hinton and Salakhutdinov 2006b)).

The subjects of a hidden operation

Given the diagrammatic forms of the basic model equations, the network diagram and the operational code comprising the privileged machine, how are subject positions assigned? The layered architecture of model generates new textures, densities, pools, sparsities and saturations in both the common vector space, and in the operations of machine learners. In early 2002, while carrying out an ethnographic study of 'extreme programming,' a software development methodology popular at that time (**???**), I spent several months visiting a company in Manchester developing software for call centres. The purpose of the software was to manage 'knowledge' in call centres such that any query from a caller would be readily answered by call centre staff who would query a knowledge management system. This system was marketed on the promise of the neural net. It contained an artificial neural network that learned to match queries and responses over time. The taciturn neural network expert, Vlad, sat in a different part of the room from the developers working on the databases and the web interfaces of the knowledge management system. His work with the

neural network was at the core of the knowledge management system yet outside the orbit of the software development team and its agile software development processes. They generally regarded Vlad and the neural net as an esoteric and temperamental yet powerful component, a hidden node we might say, of the knowledge management system. As we have already seen with `kittydar` , today the situation of neural networks has changed. They are no longer exotic or specialized, but both banal and occasionally spectacular.

How would we describe the figure of human machine learner in this setting? In *The Archaeology of Knowledge*, Michel Foucault refers to the 'position of the subject' as an anchor point for the power-laden, epistemically conditioning enunciative functions called 'statements.' When a subject position can be assigned, propositions, diagrams, numbers, models, calculations and data structures can come together in statements, as 'the specific forms of an accumulation' (Foucault 1972, 125). But this anchor point is not a unifying point grounded in interiority, in intentionality or even in single speaking position or voice (that of the machine learning expert, for instance). On the contrary, 'various enunciative modalities manifest his [sic] dispersion' (Foucault 1972, 54). In the mist of this dispersion (a dispersion that is the main focus of this chapter), the position of subject is linked to operations that determine statements that become a kind of law for the subject.

As Foucault puts it, in a formulation that effectively anticipates accounts of performativity that gain currency elsewhere several decades later,

> in each case the position of the subject is linked to the existence of an operation that is both determined and present; in each case, the subject of the statement is also the subject of the operation (he who establishes the definition of a straight line is also he who states

> it; he who posits the existence of a finite series is also, and at the same time, he who states it) ; and in each case, the subject links, by means of this operation and the statement in which it is embodied, his future statements and operations (as an enunciating subject, he accepts this statement as his own law) (Foucault 1972, 94-95).

It is fitting that Foucault's examples here include subjects who say things like 'I call straight any series of points that …,' since these are just the kinds of statements that operate in machine learning. The *operation*, however, is crucial, since the operation opens onto many different practices and techniques (function finding, optimisation, transformation of data into the common vector space, mobilisation of probability distributions as a kind of rule of existence for learnable situations, etc.) that accompany, ornament, armour and diagram the statement. But that choice of illustration is only coincidental. The more substantial point of connection here is the subject-positioning circularity that Foucault posits between the operation and accompanying statement: the subject of the statement is also the subject of the operation.

This might be loosely formalised for any machine learner as follows: the diagrammatic operations of the machine learner support the production of statements; these operations become a way of producing future statements to the extent that the subject of the operation is *also* the subject of the statement. The assignation of a subject position occurs in this forward and backward, feed-forwarding and back-propagating movement between operation and statement. Although the gap between what is done operationally and what is stated might seem small, there are many slippages and divergences in it. The specificity of seemingly minor statements such as 'we see that Net-5 does the best, having errors of only 1.6%, compared to 13% for the "vanilla" network Net-2' (Hastie, Tibshirani, and Friedman 2009, 407) bears within it, in its coupling to all the operations

comprising 'Net-5,' a set of determinations and relations for variously positioned subjects. (These might include machine learners, such as Hinton or Le Cun, but also U.S. Postal workers). In any concrete situation, in relation to any specific machine learner, the diagrammatic operations and statements will position subjects in specific ways. There is no simple referent here, no simple object facing a knowing 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.

Algorithms that propagate errors

The coincidence of the subject of the operation and the subject of the statement mobilises machine learners in the sense that it aligns future operations and statements. It puts them in movement but a form of movement propagated by specific operations. Neural nets, especially in their algorithmic operation, make this propagation between statement and operation more tangible. Take the back-propagation algorithm . In their initial publication, Rumelhart, Hinton and Williams conclude:

> The learning procedure, in its current form, is not a plausible model of learning in brains. However, applying the procedure to various tasks shows that interesting internal representations can be constructed by gradient descent in weight-space, and this suggests that it is worth looking for more biologically plausible ways of doing gradient descent in neural networks (Rumelhart, Hinton, and Williams 1986, 536).

In some respects, this is the standard disclaimer (one that was already marked in McCulloch and Pitts neuronal calculus of the 1940s): the brain has only ana-

logical relevance to machine learning, although it can certainly be a source of novel mechanisms ('biologically plausible ways of doing gradient descent'). At the same time, if a machine learner can construct 'interesting internal representations … in weight-space,' then the process whereby these representations become interesting will be of great interest. That is, when the operations of the algorithm – in this case, back-propagation – become a focus of interest or perhaps a matter of concern (Stengers 2005, 161).

How could an algorithm such as back-propagation diagram the coincidence of subject of operation and subject of statement that defines machine learners? The pivotal point here is error. Errors, error rates, training error, test error, validation error: these are just some of the errors that criss-cross between human and machine learners. Errors move between operations and statements. While not all of these errors figure directly in the algorithms, the learning procedure of most machine learners derives from the way they update model parameters in the light of incoming data, and errors or differences between expected and actual outputs. Every machine learner makes different determinations in relation to model parameters and errors. The distinctive feature of neural nets, at least in their ordinary 'vanilla' forms, consists in their use of gradient descent to minimise errors by adjusting the weights (or parameters) of all the nodes (or linear models) comprising the machine learner. Adjusting the parameters of the nodes in the neural net hardly seems a striking achievement. If we, however, look more closely at the way in which the 'representations' are iteratively constructed in neural nets, something more interesting begins to emerge from the forwards and backwards movement of this algorithm. We have already seen something of the forward movement. It is defined by the equations 1 that move data through a succession of layers and their nodes. Conversely, the equations 3 specify the way in which the weights of various nodes in the output layer nodes and the

hidden layer nodes in a neural net are updated during the back-propagation phase of the iteration:

$$\begin{aligned}\beta_{km}^{(r+1)} &= \beta_{km}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \beta_{km}^{(r)}} \\ \alpha_{ml}^{(r+1)} &= \alpha_{km}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \alpha_{km}^{(r)}}\end{aligned} \tag{3}$$

(Hastie, Tibshirani, and Friedman 2009, 396)

There are many different variables in the equations 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 data appear in N, the number of rows or observations, as well as K, the number of outputs and M, the number of nodes in the hidden layer. In the densely iconic and indexical diagram of equation 3, the interweaving of the subscripts in the two lines show that the ways in which values of the parameters in the models of the first two lines of equation 1 are varied as the model is trained on the input data. The two lines of equation 3 specify how first the values of the parameters of the K nodes of the output layer should be altered in the light of the difference between the actual and expected output values, and then how the weight of the M nodes of the hidden layer should be adjusted. Once these are adjusted, the forward movement defined by equations 1 begin 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 2. 'It is as if the error propagates from the output y back to the inputs and

hence the name *back-propagation* was coined' writes Alpaydin (Alpaydin 2010, 250). As in any gradient descent operation (see chapter **??**), 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.

If back-propagation was formulated in the 1980s (and indeed, was already known in 1960), what do we learn from its current re-iterations? Given the effort that went into crafting neural nets to recognise handwritten digits during the 1980s and 1990s, what does the revival of neural nets suggest about machine learning as feed-forward/back-propagation operation? From the early publications such as (Rumelhart, Hinton, and Williams 1985) on, the layered composition of the model has been linked to architectural considerations. As Hastie and co-authors write:

> The advantages of back-propagation are its simple, local nature. In the back propagation algorithm, each hidden unit passes and receives information only to and from units that share a connection. Hence it can be implemented efficiently on a parallel architecture computer (Hastie, Tibshirani, and Friedman 2009, 397).

These practical considerations have different significance in different settings. Some of the current iterations of neural nets in deep learning rely on massively parallel computing architectures (for instance, Andrew Ng's GoogleX Youtube video project). Yet the information sharing that happens during back-propagation might also encompass the human others of neural nets. The efficient parallel implementation in computing architecture affects, I would suggest, human and non-human machine learners in different ways. Given that equations

1 and 3 seem to function automatically, without any hand in their movement, how could humans fit here?

Competitions as examination

Again, the treatment of errors provides a useful thread to follow here in locating the subject of machine learning operations. At almost every step of its development as a field, and in almost every aspect of its operation, competitions conducted through forms of examination of error rates lie at the intersection of human and machine learners. At this intersection, errors are not purely epistemic. They circulate within a wider economy of competitive optimisation that connect them to power, value and agency. The learning of machine learning seems to incarnate learning through examinations that rank human and non-human machine learners according to error rates. We saw above that competitions to recognise handwritten digits constituted a focal point for neural nets during the 1990s. Much of the discussion of neural net in *Elements of Statistical Learning* revolves around a competition conducted in 2003. As we will soon see, machine learning competitions more generally form one of the principal ways in which people and machines come together. What can we learn from such competitions about subject positions in machine learning?

If we take a typical contemporary machine learning competition, something of the backwards and forwards movement between human and machine machine learners starts to appear. The competitions run by Kaggle exemplify many of the movements that the back-propagation algorithm implements. It efficiently implements a parallel architecture machine learning process by back-propagating errors to the 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. Unlike the

research-oriented machine learning competitions such as the annual (KDD Cup)[http://www.sigkdd.org/kddcup/index.php] run by the Association of Computing Machinery (ACM) Special Interest Group on Knowledge Discovery and Data Mining, the NIPS (Neural Information Processing Systems) Challenges, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC 2014) or the International Conference on Machine Learning (ICML)[http://machinelearning.org/icml.html], the Kaggle competitions attract a 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 entry and incorporated it into its products (**???**)). Kaggle.com is effectively an employment agency for machine learners (Kaggle 2015b). Certain competitions have recruitment as the prize (for instance, several competitions sponsored by Facebook have positions as data scientist at Facebook as the prize).

> Ever wonder what it's like to work at Facebook? Facebook and Kaggle are launching an Engineering competition for 2015. Trail blaze your way to the top of the leader board to earn an opportunity at interviewing for a role as a software engineer, working on world class Machine Learning problems (Kaggle 2015c)

While most employment agencies rely on CVs (curriculum vitae), Kaggle employs something more like back-propagation and then cross-validation between multiple competitions as a way of optimising its ranking and prospective employment of machine learners.

In Figure 3, the competition organizers list three injunctions: download (the data), build (a model), and submit (an entry or many entries to the competition). From these three movements, a variety of different outcomes appear. Leaderboards and individual rankings within the Kaggle's "world's largest community of data scientists" (Kaggle 2015d),[6] allow clients of Kaggle (corporations mostly) to 'harness the "cognitive surplus" ' (Kaggle 2015e). The figure also shows some of the typical diversity of the several hundred machine learning competitions that Kaggle has staged since 2011: diabetic retinopathy and west Nile virus prediction competition appear next to search results relevance or context ad clicks competitions. This proximity, whether accidental or constructed, between very disparate entities suggests that machine learners possess a special epistemic mobility not readily available to the experts in diabetes, virology, information retrieval or search engine optimisation. And again, the re-framing of their differences in a common vector space, subject to classification and optimisation is standard is a standard feature.

While the several hundred thousand participants in the competitions on Kaggle are all machine learners and invoke various machine learners in the models they build and submit, their participation in the competitions is itself, like nearly all the classifiers we have been discussing, subject 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 but the ranking

[6] At the time of writing, Kaggle claims around 320,000 players.

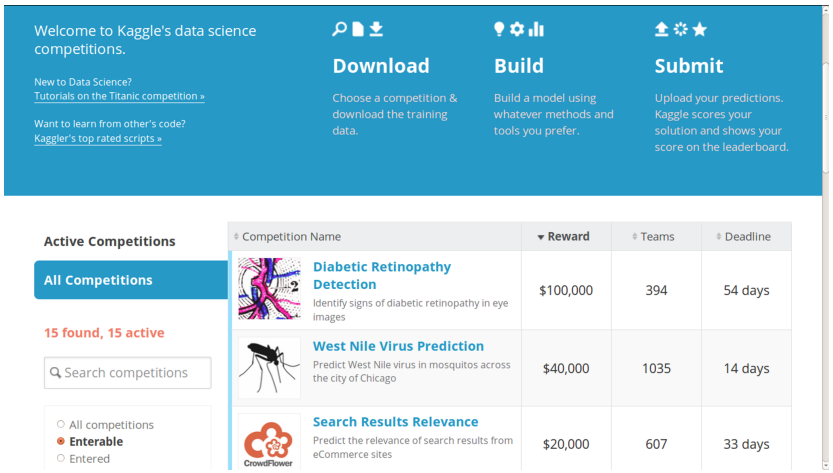


Figure 3: Kaggle data science competitions

and classification of machine learners possibly goes much deeper than this:

> The leaderboard is a central fixture of the Kaggle experience. It provides context to the incredible work accomplished by the Kaggle data science community. To a competitor, the 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 raw. It's beautiful. It's thousands of hours of collective human toil (Kaggle 2015e)

The dynamics of ranking, and the experience of being ranked here arise from a fairly simple mechanism. Entrants in a given competition download two datasets, a training dataset that includes labels for all the response variables, and a test dataset that does not include the labels. In principle, competitors

construct machine learners using the training dataset, use their machine learner to predict labels for the test dataset, and then upload the predicted labels to Kaggle as a submission to the competition. The Kaggle platform then calculates a ranking based on the generalization error in the test labels. Kaggle itself has the actual labels for the test dataset. Entrants monitor the leaderboard and attempt to improve their rankings by making new submissions with improved or altered models. The many entries that participants sometimes submit to the competitions suggest that rankings, and their visibility operate like the loss functions that many supervised machine learners use to optimise the fit of a model to the training data. Competitors optimise their entries against each other, but the competition overall functions as a kind of general optimization process in which many hidden nodes adjust their treatment to the training data as scores and rankings propagate through via the leaderboard system. The very stylised injunction to download-model-submit many times effectively creates an algorithmic process in which many hidden nodes operate in parallel to produce predictions.

It would be possible to explore in much greater ethnographic 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. The question is what happens to subject positions in machine learning competitions? The competitions all 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 in relation to the problem.

Machine learning competitions furnish a contemporary instance of the practices

of examination that Foucault described in *Discipline and Punish*:

> The examination combines the techniques of an observing hierarchy and those of a normalizing judgement. It is a normalizing gaze, a surveillance that makes it possible to qualify, to classify and to punish. It establishes over individuals a visibility through which one differentiates them and judges them. That is why, in all the mechanisms of discipline, the examination is highly ritualized. In it are combined the ceremony of power and the form of the experiment, the deployment of force and the establishment of truth. At the heart of the procedures of discipline, it manifests the subjection of those who are perceived as objects and the objectification of those who are subjected. The superimposition of the power relations and knowledge relations assumes in the examination all its visible brilliance (Foucault 1977, 183-185).

In its disciplinary form, Foucault links epistemic and operational aspects of examination. Examinations combine ceremony, ritual, experiment, force and truth, as well as processes of subjectification and objectification. If, as already been suggested above, the pack-ice of machine learning formed around neural nets during the 1990s, we might say that the re-solidification of neural nets today in high profile deep learning projects can be understood in terms of a much more pervasive practice of examining and testing occurring through machine learning training and competitions. How would such a process be legible? The forms of visibility created by competitions, the ways in which they individualize and document machine learners (often by proper names), and that maximise extractions of force, time, propensities and aptitudes. In competitive examinations, we might see attempts to construct 'parallel architectures' for people

doing machine learning. Machine learning competitions effectively implement a parallel architecture for machine learners with limited communication.

```
## Loading required package: methods
```

% latex table generated in R 3.2.0 by xtable 1.7-4 package % Thu Jul 9 11:55:35 2015

% latex table generated in R 3.2.0 by xtable 1.7-4 package % Thu Jul 9 11:55:35 2015

In many Kaggle competitions, winning entries come from teams of machine learners working together. In the National Data Science Bowl competition of 2015, the winning team comprised seven graduate and post-doctoral researchers from Ghent University, Belgium. In a jointly written blog account of their winning entry, team 'Deep Sea' describe something of the construction of the deep learning models they built. These were convolutional neural nets, neural nets in which unit of the network only 'look' at overlapping tiles of the input images:

> We started with a fairly shallow models by modern standards (~ 6 layers) and gradually added more layers when we noticed it improved performance (it usually did). Near the end of the competition, we were training models with up to 16 layers. The challenge, as always, was balancing improved performance with increased overfitting (Dieleman 2015).

Like many of the entrants in image-based classification competitions such as the ImageNet Large Scale Visual Recognition Challenge (ILSVRC 2014), 'Deep Sea' built their machine learner in several stages, first deriving features from

the data by creating various layers that looked for common features 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 tthe labels in the normal back-propagation process of neural net training.

In comparison to the neural nets discussed above, the deep belief networks involve many more parameters, stages of observation and modelling, configuration of hardware and infrastructural arrangements and comparison of results. 'Deep Sea' describe the architecture of one of their more successful models:

> It has 13 layers with parameters (10 convolutional, 3 fully connected) and 4 spatial pooling layers. The input shape is (32, 1, 95, 95), in bc01 order (batch size, number of channels, height, width). The output shape is (32, 121). For a given input, the network outputs 121 probabilities that sum to 1, one for each class.

They go on to describe the different layers – cyclic slice, convolutional, spatial pooling – that derive features from the data or augmenting it - by examining overlapping tiles, by rotating or scaling the images, so that any given image, is 'seen' in a number of different ways, and the model learns to detect these variations. (A summary of this architecture is shown in .) The combination of diverse layers in a stratified model introduces many different parameters into the model and implies heavy computational work. Crucially, the massive parallel

computing allows 'deep' learning. Infrastructure and cognition entwine heavily here, since the very possibility of building many layered neural nets depends on the possibility of relatively quickly training them. Probably many other competitors in this competition would not have had access to the Tesla K40 or 'NVIDIA GTX 980 Superclocked' GPU cards that 'Deep Sea' relied on.[7] Even with that intensive computational resource, their models required 'between 24 and 48 hours to reach convergence.' They constructed around 300 models. Because of the plethora of models with different architectures and parameters, 'we had to select how many and which models to use in the final blend. For this, we used cross-validation on our validation set' (Dieleman 2015). As is so often the case in machine learning, the configuration of machine learners by competitive examination engenders populations of models whose aggregate tendencies come to embody a model – in the sense of optimum – performance. The competition itself, then, is more like a meta-population event, since populations of models converge in a visible hierarchy within it. Both during and after the conference, the visibility of the leaderboard becomes gives access to and elicits further efforts to render visible, usually in the form of descriptions of winning entries and sometimes positing of model code.[8] The 'DeepSea' team might be a typical intersectional entity today. Like the 'wonderful people' described by Hilary Mason, they bring together infrastructure, engineering, mathematics/statistics and some knowledge of human behaviour (although the knowledge of human be-

[7] As another competitor in the National Data Science Bowl mentions:

> One example is here the Kaggle plankton detection competition. At first I thought about entering the competition as I might have a huge advantage through my 4 GPU system. I reasoned I might be able to train a very large convolutional net in a very short time – one thing that others cannot do because they lack the hardware (Dettmers 2015)

Interestingly, hardware parallelism, at least in the area of deep learning, seems to matter more than the ability to test, examine, observe or invest new model configurations.

[8] 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.'

haviour in this case might have more to do with what other Kaggle competitors might be doing, as well as an awareness of the cutting edge research leaders in the field of visual object recognition techniques).

```
l0_variable = nn.layers.InputLayer((batch_size, 1, patch_sizes[0][0], patch_sizes[0][1]
 l0c = dihedral.CyclicSliceLayer(l0_variable)

l1a = Conv2DLayer(l0c, num_filters=32, filter_size=(3, 3), border_mode="same", W=nn_plan
l1b = Conv2DLayer(l1a, num_filters=16, filter_size=(3, 3), border_mode="same", W=nn_plan
 l1 = MaxPool2DLayer(l1b, ds=(3, 3), strides=(2, 2))
 l1r = dihedral.CyclicConvRollLayer(l1)

l2a = Conv2DLayer(l1r, num_filters=64, filter_size=(3, 3), border_mode="same", W=nn_plan
l2b = Conv2DLayer(l2a, num_filters=32, filter_size=(3, 3), border_mode="same", W=nn_plan
 l2 = MaxPool2DLayer(l2b, ds=(3, 3), strides=(2, 2))
 l2r = dihedral.CyclicConvRollLayer(l2)

l3a = Conv2DLayer(l2r, num_filters=128, filter_size=(3, 3), border_mode="same", W=nn_pla
l3b = Conv2DLayer(l3a, num_filters=128, filter_size=(3, 3), border_mode="same", W=nn_pla
l3c = Conv2DLayer(l3b, num_filters=64, filter_size=(3, 3), border_mode="same", W=nn_plan
 l3 = MaxPool2DLayer(l3c, ds=(3, 3), strides=(2, 2))
 l3r = dihedral.CyclicConvRollLayer(l3)

l4a = Conv2DLayer(l3r, num_filters=256, filter_size=(3, 3), border_mode="same", W=nn_pla
l4b = Conv2DLayer(l4a, num_filters=256, filter_size=(3, 3), border_mode="same", W=nn_pla
l4c = Conv2DLayer(l4b, num_filters=128, filter_size=(3, 3), border_mode="same", W=nn_pla
 l4 = MaxPool2DLayer(l4c, ds=(3, 3), strides=(2, 2))
 l4r = dihedral.CyclicConvRollLayer(l4)
```

```
 l4f = nn.layers.flatten(l4r)

l5 = nn.layers.DenseLayer(l4f, num_units=256, W=nn_plankton.Orthogonal(1.0), b=nn.init.
 l5r = dihedral.CyclicRollLayer(l5)

l6 = nn.layers.DenseLayer(l5r, num_units=256, W=nn_plankton.Orthogonal(1.0), b=nn.init.
l_variable = dihedral.CyclicPoolLayer(l6, pool_function=nn_plankton.rms)



 # fixed scale part
l0_fixed = nn.layers.InputLayer((batch_size, 1, patch_sizes[1][0], patch_sizes[1][1]))
 l0c = dihedral.CyclicSliceLayer(l0_fixed)

l1a = Conv2DLayer(l0c, num_filters=16, filter_size=(3, 3), border_mode="same", W=nn_plan
l1b = Conv2DLayer(l1a, num_filters=8, filter_size=(3, 3), border_mode="same", W=nn_plank
 l1 = MaxPool2DLayer(l1b, ds=(3, 3), strides=(2, 2))
 l1r = dihedral.CyclicConvRollLayer(l1)

l2a = Conv2DLayer(l1r, num_filters=32, filter_size=(3, 3), border_mode="same", W=nn_plan
l2b = Conv2DLayer(l2a, num_filters=16, filter_size=(3, 3), border_mode="same", W=nn_plan
 l2 = MaxPool2DLayer(l2b, ds=(3, 3), strides=(2, 2))
 l2r = dihedral.CyclicConvRollLayer(l2)

l3a = Conv2DLayer(l2r, num_filters=64, filter_size=(3, 3), border_mode="same", W=nn_plan
l3b = Conv2DLayer(l3a, num_filters=64, filter_size=(3, 3), border_mode="same", W=nn_plan
l3c = Conv2DLayer(l3b, num_filters=32, filter_size=(3, 3), border_mode="same", W=nn_plan
 l3 = MaxPool2DLayer(l3c, ds=(3, 3), strides=(2, 2))
```

```
    l3r = dihedral.CyclicConvRollLayer(l3)
    l3f = nn.layers.flatten(l3r)

l4 = nn.layers.DenseLayer(l3f, num_units=128, W=nn_plankton.Orthogonal(1.0), b=nn.init.
    l4r = dihedral.CyclicRollLayer(l4)

l5 = nn.layers.DenseLayer(l4r, num_units=128, W=nn_plankton.Orthogonal(1.0), b=nn.init.
l_fixed = dihedral.CyclicPoolLayer(l5, pool_function=nn_plankton.rms)



    # merge the parts
    l_merged = nn.layers.concat([l_variable, l_fixed])

l7 = nn.layers.DenseLayer(l_merged, num_units=data.num_classes, nonlinearity=T.nnet.sof
```

In all of this augmentation, pre-processing, unsupervised and supervised training, the competitions accelerate the travel of machine learners across the terrain of science, industry, media and government. For the deep learners, many problems of classification and recognition start to look like handwritten digits. What the neural nets perhaps most strongly diagrammatically show is how the act of seeing machine learning itself. Hinton's references to 'mind' might make sense here as part of the re-drawing human-machine differences. If some machine learners, such as neural nets, afford visibility to what they see in the world, then their function as 'minds,' as cognitive processes – I think this is what Hinton has in mind as a cognitive psychologist – effectively does begin to internalise the operations and statements that previously were handled by human machine learners. This re-distribution does privilege the machine as site of learning, but also quickly assigns new subject positions. The members of 'DeepSea' build

models that can accurately classify more than a hundred kinds of plankton in short order. They are the 'wonderful people' that Hilary Mason describes . But they also occupy the subject position assigned by the conjunction between the operation and the statements entailed in machine learning. Note the punchline here is: the neural nets in the form of deep belief nets are automating through their many convolutional layers the feature engineering that characterised the skilled configuration work in machine learning in decision trees, linear regressions, support vector machines and so forth.

Conclusion

Very late in this book, it seems, we finally reach people doing things. Why so late? Neural nets figure the vicissitudes of human machine learners quite well. They test machine learners for their capacity to understand how models relate to data, and they test them in relation to their knowledge and experience of how of to craft and to regularize the parameters of models in a given situation. Perhaps more profoundly, they figure a much more deeply competitive imperative that frames much of the practice in machine learning, and in many ways constrains thinking around machine learning. This competition is not always explicit or overt but it almost transpires in the form of examination.

In *Doing Data Science*, Cathy O'Neil and Rachel Schutt offer a pragmatic set of steps for working with machine learners:

> The big picture is that given data, a real-world classification problem, and constraints, you need to determine: >1. Which classifier to use >2. Which optimization method to employ >3. Which loss function to minimize >4. Which features to take from the data >5. Which evaluation metric to use

[@Schutt_2013, 116]

Perhaps the order of these steps would be contested by some people, but each of these determinations is definitely a matter of competitive contention between machine learners. Competitions encourage a proliferation of determinations. Whether or not this competition deeply structures the field of machine learning is also a matter for further investigation. But in my experience, immersion in the field of machine learning, navigating its social textures, and getting a feeling for how it is to exist in these conditions definitely entails a sense of becoming the subject of examination. This feeling, which is concretised in the numerous machine learning competitions and online instructional materials, is convoluted because it concerns the constitution of machine learners themselves as operations of examination and as a form of documentation or statement of examination, test, and validation focused on errors. As Andrew Ng opines, 'we care about generalization error,' , but error here both refers to something that neither the responsibility simply of the machine or its human others. Rather, responses to and care for this error move forwards and backwards between the human and non-human machine learners.

Throughout this chapter, the proximities between the human and machine machine learners have been discussed in two different ways. First, in terms an intersection that assigns a subject position through a machine learning operation. Neural nets epitomise this operation. The subject who operates the neural net as a way of fitting a model has been increasingly caught up in a network of neural nets that feed forward into a widening stream of parallel and layered architectures. This feeding-forward, however, is regularized or narrowed down through examination and error, through back-propagation on various scales. The principle of testing through generalization error that has long guided the supervision of machine learners has been generalized to an examination or test

of machine learners in competitions. While competitive testing and examination has long been a constitutive practice in machine learning (for instance, in the case of the handwritten digit recognition contests of the 1990s), these competitions have now become a somewhat general mechanism for assigning positions to machine learners. The positions assigned include the subject positions of people whose participation in machine learning competitions are ranked and indeed the machines themselves, whose rankings rise and fall as different infrastructures, algorithms and optimisations come into play.

Neural nets figure this re-iterative drawing of human-machine learning differences in several respects. Their own ups and downs, their potential to drive down error or learn features from data given enough data derives less from some exotic mathematical abstraction or encompassing algorithm, and more from an accumulation of layers and connections between units of modelling. The central algorithm – back-propagation – is instructive here: it propagates errors throughout all the elements of the network, and every element in the network adjusts its weights in trying to minimise error. The predictive power of the model derives from the networked collective of linear models all seeking to optimise their error rates. So too, the competitive examinations that constitute and today generalize machine learning as a practice predicate the ongoing potential of hidden units – machine learners – to learn from their rankings in tests of error, and to collectively learn from those rankings.

What does this algorithmic propagation and generalization of errors and their testing mean for subject positions associated with machine learning? The cultural theorist Lauren Berlant describes optimism as an operation:

> The surrender to the return to the scene where the object hovers in its potentialities is the operation of optimism as an affective form

(Berlant 2007, 20)

This is a complicated formulation since it brings together surrender, return, scene, object, potentialities and affective form. All of these movements, places and things might be understood as purely psychic or even semiotic. I would suggest that optimism as an asignifying diagrammatic operation also plays out across the manifold surfaces of algorithms, datasets, models, platforms and ranking systems associated with machine learning as a competitive examination. The 'object hovers in its potentialities' in the case of machine learning because neural nets and their kin assimilate and adjust their weights in response to changes in infrastructures and in the generalization of operations to newly adjacent domains. Machine learners generate optimism through and about optimisation, an optimisation that is deeply predictive, prospective, anticipatory and abductive. But this adjusting of weights carried out through the propagation of errors is also inherently a ranking or examination.

The Kaggle competitions play out the logic of the operation of optimism most directly. The leaderboard is an examinatory ranking of efforts to drive down error rates, the same efforts that perhaps first affected machine learning around handwritten digit recognition in the 1990s. Entrants work hard to optimise the generalization errors of their models, but Kaggle aligns their work in a bootstrap aggregating architecture that assigns subject positions based around those error rates. Maurizio Lazzarato describes how subject positions are assigned through examination, testing and ranking:

> The thought and consciousness of the individuated subject come into play when there is an obstacle, a disturbance, or an "event." Then the subject, consciousness, and representation are used in order to modify the feedback relations among the human and non-human

> components of the automobile "machine," in order to reestablish the automatic mechanisms and machinic operations (Lazzarato 2014, 89-90)

While the ensembles of machine learners we saw in the Kaggle competitions are not entirely individuated through the competitions, the Kaggle competitions certainly help establish new automatic and machinic operations. The ranking of machine learners (human and non-human) then should be understood as less a privileging of wonderful people at the intersection of infrastructure, human behaviour and mathematics and more as a re-iterative drawing of distinctions between people that at the same time modifies feed-forward and feed-back relations between people and machines. These relations almost always prospectively marginalize humans, even as they also inaugurate ontological mutations.

References

Ackley, David H., Geoffrey E. Hinton, and Terrence J. Sejnowski. 1985. "A learning algorithm for Boltzmann machines." *Cognitive science* 9 (1): 147–69. http://www.sciencedirect.com/science/article/pii/S0364021385800124.

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

Bacon. 2012. "Hilary Mason - Machine Learning for Hackers." http://vimeo.com/43547079.

Berlant, L. 2007. "Nearly Utopian, Nearly Normal: Post-Fordist Affect in La Promesse and Rosetta." *Public Culture* 19 (2): 273. http://lucian.uchicago.edu/blogs/politicalfeeling/files/2007/10/berlant-la-promesse.pdf.

Bishop, Christopher M., and Nasser M. Nasrabadi. 2006. *Pattern recognition*

and machine learning. Vol. 1. springer New York. http://www.library.wisc.edu/selectedtocs/bg0137.pdf.

Bishop, Christopher M., and others. 1995. "Neural Networks for Pattern Recognition." http://www.engineering.upm.ro/master-ie/sacpi/mat_did/info068/docum/Neural/%20Networks/%20for/%20Pattern/%20Recognition.pdf.

CNN. 2011. "40 Under 40: Ones to Watch. CNNMoney." http://money.cnn.com/galleries/2011/news/companies/1110/gallery.40_under_40_ones_to_watch.fortune/.

Dahl, George. 2013. "Deep Learning How I Did It: Merck 1st place interview. no free hunch." http://blog.kaggle.com/2012/11/01/deep-learning-how-i-did-it-merck-1st-place-interview/

Dettmers, Tim. 2015. "Which GPU(s) to Get for Deep Learning: My Experience and Advice for Using GPUs in. Deep Learning." https://timdettmers.wordpress.com/2014/08/14/which-gpu-for-deep-learning/.

Dieleman, Sander. 2015. "Classifying Plankton with Deep Neural Networks. Sander Dieleman." http://benanne.github.io/2015/03/17/plankton.html.

Edwards, Paul N. 1996. *The Closed World : Computers and the Politics of Discourse in Cold War*. Inside Technology. Cambridge, Mass. ; London: MIT Press.

Foucault, Michel. 1972. "The Archaeology of Knowledge and the Discourse on Language (Trans: A. Sheridan)." *Pantheon, New York*.

———. 1977. *Discipline and Punish: The Birth of the Prison*. Translated by Allan Sheridan. New York: Vintage.

Fritsch, Stefan, and Frauke Guenther. 2012. *Neuralnet: Training of Neural Networks*. http://CRAN.R-project.org/package=neuralnet.

Gomes, Lee. 2014. "Machine-Learning Maestro Michael Jordan on the Delu-

sions of Big Data and Other Huge Engineering Efforts - IEEE Spectrum." http://spectrum.ieee.org/robotics/artificial-intelligence/machinelearning-maestro-michael-jordan-on-the-delusi

Halpern, Orit. 2015. *Beautiful Data*. Durham, N.C: Duke University Press.

Hassabis, Demis, R. Nathan Spreng, Andrei A. Rusu, Clifford A. Robbins, Raymond A. Mar, and Daniel L. Schacter. 2013. "Imagine All the People: How the Brain Creates and Uses Personality Models to Predict Behavior." *Cerebral Cortex*, bht042. http://cercor.oxfordjournals.org/content/early/2013/03/04/cercor.bht042.short.

Hastie, Trevor, Robert Tibshirani, and Jerome H. Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd edition. New York: Springer.

Hinton, Geoffrey E. 1989. "Connectionist Learning Procedures." *Artificial Intelligence* 40 (1): 185–234. http://www.sciencedirect.com.ezproxy.lancs.ac.uk/science/article/pii/0004370289900490.

Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. 2006a. "Reducing the Dimensionality of Data with Neural Networks." *Science* 313 (5786): 504–7. http://www.sciencemag.org/content/313/5786/504.short.

———. 2006b. "Reducing the dimensionality of data with neural networks." *Science* 313 (5786): 504–7. http://www.sciencemag.org/content/313/5786/504.short.

Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. 2006. "A Fast Learning Algorithm for Deep Belief Nets." *Neural Computation* 18 (7): 1527–54. http://www.mitpressjournals.org/doi/abs/10.1162/neco.2006.18.7.1527.

Hof, Robert D. 2014. "Chinese Search Giant Baidu Thinks AI Pioneer Andrew Ng Can Help It Challenge Google and Become a Global Power. MIT Tech-

nology Review." http://www.technologyreview.com/featuredstory/530016/a-chinese-internet-giant-starts-to-dream/.

ILSVRC. 2014. "ImageNet Large Scale Visual Recognition Competition 2014 (ILSVRC2014)." http://www.image-net.org/challenges/LSVRC/2014/.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An Introduction to Statistical Learning*. Springer. http://link.springer.com/content/pdf/10.1007/978-1-4614-7138-7.pdf.

Kaggle. 2015a. "Facebook Recruiting IV: Human or Robot? | Kaggle." https://www.kaggle.com/c/facebook-recruiting-iv-human-or-bot.

———. 2015b. "Description - Leaping Leaderboard Leapfrogs | Kaggle." https://www.kaggle.com/c/leapfrogging-leaderboards.

———. 2015c. "About | Kaggle." https://www.kaggle.com/about.

———. 2015d. "Data Science Jobs Forum | Kaggle." https://www.kaggle.com/jobs.

———. 2015e. "Description - Facebook Recruiting IV: Human or Robot? | Kaggle." https://www.kaggle.com/c/facebook-recruiting-iv-human-or-bot.

KDD. 2013. "Call For KDD Cup." http://www.kdd.org/kdd2013/call-for-cup.

Kirk, Matthew. 2014. *Thoughtful Machine Learning: A Test-Driven Approach*. 1 edition. Sebastopol, Calif.: O'Reilly Media.

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

Lazzarato, Maurizio. 2014. *Signs and Machines: Capitalism and the Production of Subjectivity*. Semiotext (e). http://mitpress.mit.edu/books/signs-and-machines.

"Lecture 1 | Machine Learning (Stanford). Youtube." 2008. http://www.

youtube.com/watch?v=UzxYlbK2c7E/&feature=youtube_gdata_player.

"Lecture 6 | Machine Learning (Stanford)." 2008. http://www.youtube.com/watch?v=qyyJKd-zXRE/&feature=youtube_gdata_player.

"Lecture 9 | Machine Learning (Stanford)." 2008. https://www.youtube.com/watch?v=tojaGtMPo5U/&feature=youtube_gdata_player.

LeCun, Yann, and Corinna Cortes. 2012. "MNIST Handwritten Digit Database, Yann LeCun, Corinna Cortes and Chris Burges." http://yann.lecun.com/exdb/mnist/.

LeCun, Yann, Bernhard Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne Hubbard, and Lawrence D. Jackel. 1989. "Backpropagation Applied to Handwritten Zip Code Recognition." *Neural Computation* 1 (4): 541–51. http://www.mitpressjournals.org/doi/abs/10.1162/neco.1989.1.4.541.

Mackenzie, Adrian. 2013. "'Wonderful People': Programmers in the Regime of Anticipation." *Subjectivity* 6 (4): 391–405.

Markoff, John. 2012. "In a Big Network of Computers, Evidence of Machine Learning." *The New York Times*, June. http://www.nytimes.com/2012/06/26/technology/in-a-big-network-of-computers-evidence-of-machine-learning.html.

Mason, Hilary. 2012. "Hilary Mason - Machine Learning for Hackers." http://vimeo.com/43547079.

McClelland, James L., and David E. Rumelhart. 1986. *Parallel Distributed Processing. Explorations in the Microstructure of Cognition.* Vol. 1. Cambridge, MA & London: MIT Press.

Minsky, Marvin, and Seymour Papert. 1969. "Perceptron: an introduction to computational geometry." *The MIT Press, Cambridge, expanded edition* 19: 88.

Richert, Willi, and Luis Pedro Coelho. 2013. *Building Machine Learning Systems with Python*. Birmingham: Packt Publishing.

Ripley, Brian. 1996. *Pattern Recognition and Neural Networks. 1996*. Cambridge ; New York: Cambridge University Press.

Rosenblatt, F. 1958. "The perceptron: A probabilistic model for information storage and organization in the brain." *Psychological Review* 65 (6): 386–408. doi:10.1037/h0042519.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. 1985. *Learning Internal Representations by Error Propagation*. DTIC Document. http://oai.dtic.mil/oai/oai?verb=getRecord/&metadataPrefix=html/&identifier=ADA164453.

———. 1986. "Learning Representations by Back-Propagating Errors." *Nature* 323 (6088): 533–36. doi:10.1038/323533a0.

Schutt, Rachel, and Cathy O'Neil. 2013. *Doing Data Science*. Sebastopol, Calif.: O'Reilly & Associates Inc.

Stengers, Isabelle. 2005. "Deleuze and Guattari's Last Enigmatic Message." *Angelaki* 10 (1): 151–67.

Suchman, Lucy. 2006. *Human and Machine Reconfigurations: Plans and Situated Actions*. 2nd ed. Cambridge University Press.

Wilson, Elizabeth A. 2010. *Affect and Artificial Intelligence*. University of Washington Press.

Zare, Douglas. 2012. "Difference Between Logistic Regression and Neural Networks - Cross Validated. CrossValidated." http://stats.stackexchange.com/questions/43538/difference-between-logistic-regression-and-neural-networks.

	domain	data_type
1	entertainment	preferences
2	entertainment	events
3	entertainment	events
4	medicine	measurements
5	tourism	events
6	finance	prices
7	entertainment	rankings
8	tourism	events
9	social_media	network
10	software	actions
11	traffic	times
12	education	texts
13	traffic	actions
14	publication	texts
15	entertainment	events
16	disaster	actions
17	science	images
18	publication	actions
19	retail	preferences
20	insurance	attributes
21	learning	images
22	media	images
23	finance	attributes
24	traffic	attributes
25	finance	events
26	learning	other
27	education	actions
28	entertainment	actions
29	identification	actions
30	publication	texts
31	environment	measurements
32	education	texts
33	finance	events
34	social_media	actions
35	advertising	actions

	Layer.type	Size	Output.shape
1	cyclic slice		(128, 1, 95, 95)
2	convolution	32 3x3 filters	(128, 32, 95, 95)
3	convolution	16 3x3 filters	(128, 16, 95, 95)
4	max pooling	3x3, stride 2	(128, 16, 47, 47)
5	cyclic roll		(128, 64, 47, 47)
6	convolution	64 3x3 filters	(128, 64, 47, 47)
7	convolution	32 3x3 filters	(128, 32, 47, 47)
8	max pooling	3x3, stride 2	(128, 32, 23, 23)
9	cyclic roll		(128, 128, 23, 23)
10	convolution	128 3x3 filters	(128, 128, 23, 23)
11	convolution	128 3x3 filters	(128, 128, 23, 23)
12	convolution	64 3x3 filters	(128, 64, 23, 23)
13	max pooling	3x3, stride 2	(128, 64, 11, 11)
14	cyclic roll		(128, 256, 11, 11)
15	convolution	256 3x3 filters	(128, 256, 11, 11)
16	convolution	256 3x3 filters	(128, 256, 11, 11)
17	convolution	128 3x3 filters	(128, 128, 11, 11)
18	max pooling	3x3, stride 2	(128, 128, 5, 5)
19	cyclic roll		(128, 512, 5, 5)
20	fully connected	512 2-piece maxout units	(128, 512)
21	cyclic pooling (rms)		(32, 512)
22	fully connected	512 2-piece maxout units	(32, 512)
23	fully connected	121-way softmax	(32, 121)