Can statistics help us to understand deep learning?
Literature Review

Johannes Smit

December 2018

1 Introduction

Since the turn of the millennium, machine learning, and particularly deep learning have been able to solve many problems that were once thought impossible for a computer. Machines can now identify objects (Li, Katz and Culler 2018), recommend videos (Bennett, Lanning *et al.* 2007), beat humans at PSPACE-hard games such as Go (Chao *et al.* 2018) and even drive cars (Gerla *et al.* 2014). These tasks are so complex that designing an algorithm by hand to tackle them would be impossible, so instead, we create a set of instructions to build an algorithm, which a computer follows. One type of algorithm is the artificial neural network, which takes inspiration from the neural structure of biological brains. Artificial neural networks consist of units called 'neurons', individually they perform a simple nonlinear operation, but when connected, a network of neurons can learn complex nonlinear patterns. Due to the abstracted nature of a machine learning algorithm's calculations, it can be difficult to provide a human understandable explanation for its reasoning, and often it is impossible. 'While an individual neuron may be understood, and clusters of neurons' general purpose vaguely grasped, the whole is beyond. Nonetheless, it works' (Grey 2017).

These algorithms are integrated further into society and are made to evaluate more real world decisions, the ability to understand what is happening inside the 'black box' becomes more important. As autonomous vehicles are becoming more ubiquitous, we will be faced with edge cases which no

human could have predicted. In cases where a driverless car does something unforeseen, it will be useful to have some understanding of what the algorithm was 'thinking' when it made the decision. Machine learning systems which aid in probation and parole decisions in the U.S. are now also being used for sentencing, but have come under a lot of criticism for being racially biased (Christin, Rosenblat and Boyd 2015). The question of how an algorithm can be 'held accountable' has been raised (Christin, Rosenblat and Boyd 2015, p. 9), but it is not known what accountability looks like for an unintelligible artificial mind.

In this literature review, I will discuss the history of the field of machine learning, and then describe the idea of a Gaussian process, which has been proposed as a way to see inside the 'black box' of deep learning.

2 Deep learning

2.1 Machine learning

We are in an age where there is so much data that in many cases, hand building a model to analyse, find patterns in and draw conclusions from the data is unfeasible, and so we use computers to analyse it in a process called 'Machine Learning' (ML) (Murphy 2012, p. 1). Computers are able to do these tasks by combining information in nonlinear ways, but become more powerful when the output of one nonlinear process is fed into another nonlinear process, abstracting the raw data to a higher level. 'With the composition of enough such transformations, very complex functions can be learned' (LeCun, Bengio and Hinton 2015, p. 436).

2.1.1 Types of learning

In 'unsupervised learning', the algorithm is given only input data and its goal is to learn the patterns that make up the raw data. This is most commonly used for cluster analysis, where the goal is to group unlabelled datapoints in a way that points with similar properties are in the same class. For example, in market basket analysis, an ML algorithm will analyse the items in a customer's online basket and suggest similar products that the customer is likely to buy.

In 'supervised learning', the algorithm is given a 'training set' of inputs and their corresponding outputs. An error score is calculated to evaluate the success of the predictions, which the algorithm tries to minimise (LeCun, Bengio and Hinton 2015, p. 436). The model learns the patterns of the training data, and then tries to predict the outcomes of some unseen 'test set' which checks if the algorithm has merely 'memorised' the patterns or if it has discovered some underlying structure in the data (LeCun, Bengio and Hinton 2015, pp. 463–467).

A compromise between these two methods is 'semi-supervised learning' (or 'partially supervised learning'), where there is a high cost of labelling the training set, so only a portion of the data are labelled. This is used in areas such as object recognition, where a human must label the objects at first, but once the algorithm can reliably identify objects in the images, it can become unsupervised.

Another very different type of learning is 'reinforcement learning', where the algorithm is an agent that must decide on actions to take without knowing the consequences. It receives a reward or loss based on its actions, and then updates its future decisions accordingly (Alom *et al.* 2018, p. 2). An example might be a robot navigating a maze. It takes a loss for touching hazards and is rewarded for reaching a target location. The key differences with this method are that the algorithm cannot directly see the function it is trying to optimise, and any action it takes will change its state.

2.1.2 Types of prediction

Supervised ML has two common prediction types: regression, where the outputs are continuous values; and classification, where the aim is to correctly assign the points to classes. Classification can either involve predicting the most likely class for a datapoint or estimating the probability of being in each class, which is equivalent to regression of $y_1, y_2, \ldots, y_n$, where $y_i \in [0, 1]$ is the probability of being in class i.

2.2 The neuron

In 1957, psychologist Frank Rosenblatt proposed a stochastic electronic brain model, which he called a 'perceptron' (Rosenblatt 1957). At the time,

most models of the brain were deterministic algorithms which could recreate a single neural phenomenon such as memorisation or object recognition. Rosenblatt's biggest criticism of these models was that while a deterministic algorithm could perform a single task perfectly, unlike a biological brain, it could not be generalised to perform many tasks without a lot of substantial changes. He described deterministic models of the brain as 'amount[ing] simply to logical contrivances for performing particular algorithms [...] in response to sequences of stimuli' (Rosenblatt 1958, p. 387). Another way he wanted his synthetic brain to mirror biological brains was the property of redundancy, the ability to have pieces removed and still function, which is not possible for deterministic algorithms where even a small change to a circuit or a line of code can stop all functionality. It was a commonly held belief that deterministic algorithms 'would require only a refinement or modification of existing principles' (Rosenblatt 1958, p. 387), but Rosenblatt questioned this idea, believing that the problem needed a more fundamental re-evaluation. At the heart of his idea was the 'perceptron', which could – through repeated training and testing – receive a set of inputs and reliably give a correct binary response. The perceptron was later generalised to the concept of an '(artificial) neuron' (also called a 'unit'), which, instead of only giving a binary output, maps a finite number of real inputs to a single real output value.

Since the invention of Rosenblatt's perceptron, there have been many alternative models for neurons proposed, including fuzzy neurons, which allow truth values to be on a scale rather than a Boolean value (Gupta and Qi 1991); higher order neurons, which include second or higher products of the inputs (Rumelhart, Hinton and McClelland 1986) and spiking neural networks, which use differential equations to emulate the electrical charge of biological neurons (Maass 1997). We will only consider the simple neuron based on Rosenblatt's perceptron.

A neuron is defined by its weight vector $\boldsymbol{w}$, its bias b and its activation function $\phi(\,\cdot\,)$.

The stages of a neuron, seen in Figure 1 are:

1. For an input vector $\boldsymbol{x} \in \mathbb{R}^n$ and a weight vector $\boldsymbol{w} \in \mathbb{R}^n$, take a

weighted sum of the inputs, called a 'linear combiner'.

$$\text{linear combiner} = \sum_{i=1}^{n} x_i w_i$$

2. Then, a bias $b \in \mathbb{R}$ is added, which translates the output to a suitable range, called the pre-activation (v). The importance of adding a bias is most clearly seen in the design of Rosenblatt's original perceptron using electronic circuitry, where a positive pre-activation would induce a current in the output wire, and a negative output would not. The bias controls how large the weighted sum needs to be to 'activate' the neuron.

$$\text{pre-activation} = v = \sum_{i=1}^{n} x_i w_i + b$$

3. Finally, an 'activation function' (or 'limiter function') $\phi(\,\cdot\,)$ is applied, which restricts the output to a range and introduces nonlinearity to the system. As mentioned, Rosenblatt's perceptrons used the $\text{sign}(\,\cdot\,)$ function, which is now only used for binary classification problems. It is not useful for connecting to another neuron, as information about the magnitude of the output is lost. Commonly the sigmoid function $S(x) = (1 + \exp(-x))^{-1}$, the $\tanh(\,\cdot\,)$ function or the 'softmax' function (a generalisation of the logistic function to vectors) are used. The most commonly used activation function for deep learning (see Section 2.6) is the Rectified Linear Unit (ReLU) function (Ramachandran, Zoph and Le 2017), which is defined as the positive part of its input $\text{ReLU}(x) = \max(0, x)$. If the activation function is the identity function, then optimising a neuron is equivalent to performing linear regression. The activation function must be differentiable if the gradient descent method is used (see Section 2.4) as gradient descent relies on calculating the gradient of the error.

$$\text{neuron output} = y = \phi\left(\sum_{i=1}^{n} x_i w_i + b\right)$$

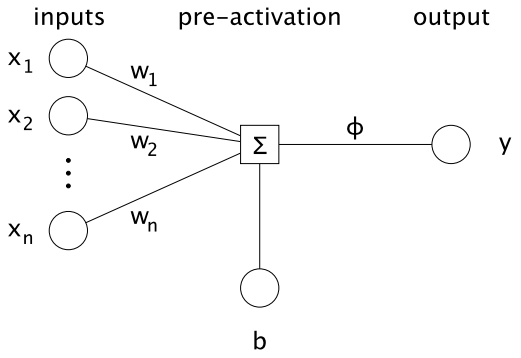


Figure 1: A diagram of the steps in a neuron.

2.3 Artificial neural networks

While a single perceptron can solve simple tasks, they become much more effective when connected together in a 'neural network' (or Artificial Neural Network (ANN)). We will only consider 'feedforward neural networks', where the neurons are connected in layers. The first and last layers are known as the input and output layers respectively, and the layers in between are called 'hidden layers'. In feedforward networks such as Figure 2, the neurons on each layer are only connected to the neurons on adjacent layers and are not connected to each other. This means information is passed through the network sequentially, as opposed to designs such as 'recurrent neural networks', where the information is recursively passed in a loop before being output.

2.4 The backpropagation algorithm

Even in a small neural network like in Figure 3, there are more than 60 weights and biases which need to be tuned to produce accurate predictions. A method to automatically optimise this was not understood until the 1980s, when the 'backpropagation' algorithm was independently discovered by Rumelhart, Hinton and Williams (1986) among others.

To understand the backpropagation algorithm, consider a network with only one output neuron (such as in Figure 3), and consider its prediction for

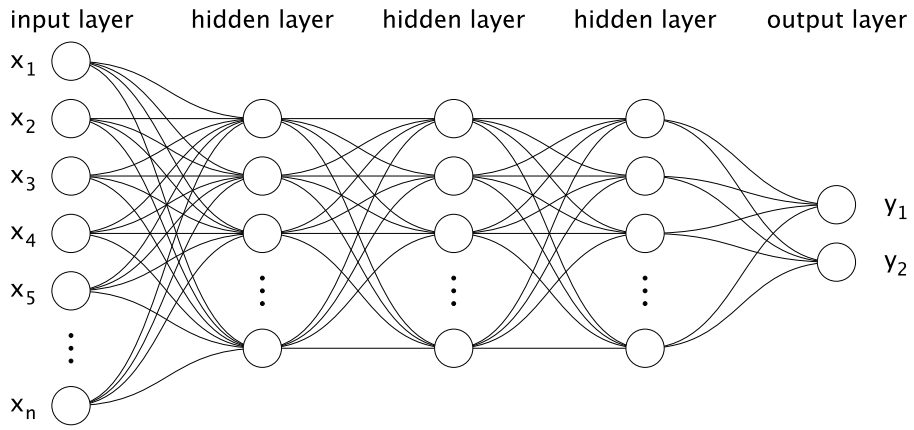


Figure 2: An example of a neural network's structure with n inputs and two outputs.

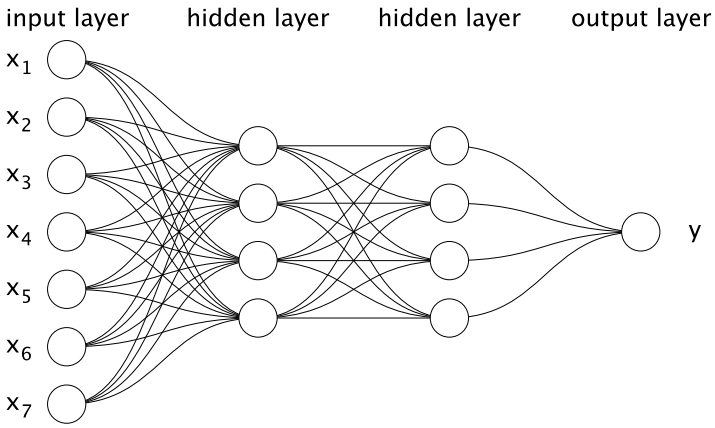


Figure 3: A neural network.

a specific datapoint in the training set. The squared error for this datapoint is $\varepsilon = \left(y^{\text{target}} - y^{\text{prediction}}\right)^2$. The target is fixed, and we cannot directly change the prediction, so to reduce ε, we can calculate the change we need to make to each weight to reduce the error the most. The prediction only depends on the outputs of the neurons on the previous layer and the weights connecting the layers. This same logic applies to all of the neurons in the network. Assuming that all the activation functions are differentiable, we can apply the chain rule repeatedly to calculate the gradient of the overall error with respect to each weight w_j in the network. This gradient

$$\Delta w_j = \frac{\partial \varepsilon(w_j)}{\partial w_j}$$

represents the change needed to weight w_j to most steeply increase the error, so the value $-\Delta w_j$ is the direction that most steeply decreases the error. We scale this amount by a constant hyperparameter η, called the 'learning rate', and change each weight accordingly $w_j = w_j - \eta \Delta w_j$. We perform this process for each weight, and then for each datapoint in the training dataset.

This type of method is called 'stochastic gradient descent' (also called 'online gradient descent' or 'iterative gradient descent') because it does not calculate the true steepest gradient. Each datapoint moves the weights in a slightly different direction, so the optimisation process zigzags towards the optimum, rather than taking the steepest path. An alternative is 'batch gradient descent', where the sum of the squared errors for all the training data is calculated and optimised.

$$\varepsilon(\boldsymbol{w}) = \frac{1}{2} \sum_i \left(y_i^{\text{target}} - y_i^{\text{output}}\right)^2$$

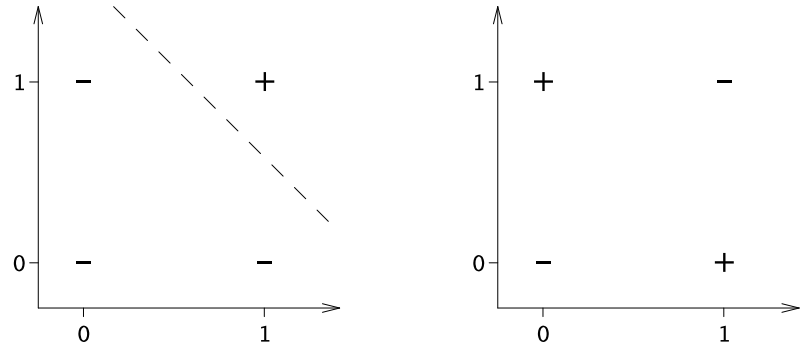
This will converge to a solution in fewer steps, but takes significantly longer to compute.

Estimating the parameters of an ANN using either type of gradient descent is guaranteed to find a local optimum given enough time, but is not guaranteed to converge to a global optimum. It is quite rare for the algorithm to get trapped in a local minimum, but a more common problem is getting stuck at saddle points where the gradient is zero (LeCun, Bengio and Hinton 2015, p. 438).

2.5 Minsky and Papert and the 'AI winter'

In 1969, Minsky and Papert published 'Perceptrons'[1], in which they described some of the limitations of Rosenblatt's perceptrons (Minsky and Papert 1987). They admitted that the perceptron appears to be a powerful tool due to properties like the 'perceptron convergence theorem', which states that for any linearly separable dataset, the perceptron classifier algorithm will find a solution in a finite number of steps. However, their main criticism was that a perceptron could not solve linearly inseparable problems, which are problems where it is not possible to draw a single straight line, plane or hyperplane that correctly divides the points into their classes.

The example they used was the XOR function, defined as being true only when only one of its inputs is true (seen in Figure 4b), which they proved was impossible for the perceptron to learn.



(a) The AND function is linearly separable.

(b) The XOR function is not linearly separable.

Figure 4: Two functions, the four possible input combinations and their outcomes. For the XOR function, there is no straight line that divides the two classes, so the XOR function is not linearly separable.

The biggest problem with Minsky and Papert's criticisms were that they only considered the 'simple perceptron', a network with only one layer, and did not look at the possibility of multiple perceptrons being joined together (Block 1970, p. 514). Minsky and Papert seemed to misunderstand Rosenblatt's reasoning for developing the perceptron. Their goal was

[1]Not to be confused with 'The Perceptron' by Rosenblatt (1957)

to disprove the perceptron as a theory for the internals of the brain, however, Rosenblatt was more interested in recreating certain properties of a biological brain without regard for about the accuracy of the internal mechanisms (Block 1970, p. 516). Despite the name of their book being 'Perceptrons', they studied 'a severely limited class of machines from a viewpoint quite alien to Rosenblatt's' (Block 1970, p. 517).

As with many promising technologies, researchers working on perceptrons overpromised and underdelivered, causing a decline in interest and funding for the field. For ML and ANNs, this period occurred in the 1970s, where other methods could produce similar results to a multilayer perceptron for the same or a much lower computing cost. This period has been called the 'AI winter'. The release of 'Perceptrons' is often blamed with causing this fallow period in funding for ML, but Minsky and Papert claim that progress was already stagnant when they published their criticisms, and it would have occurred irrespective of their opinions. With the revival of neural networks in the 1980s, it has been proposed that the severity of Minsky and Papert's criticisms as the cause of abandonment of ANNs was exaggerated to help legitimise alternative disciplines (Olazaran 1996, p. 649).

2.6 Deep learning and the 21st century

As computing power has increased exponentially (following Moore's law), ANNs have become a more viable method of prediction. In the 21st century, success has been found in increasing the number of layers in the network (depth) rather than the number of neurons in each layer (width). These 'deep neural networks' 'can implement extremely intricate functions of its inputs that are simultaneously sensitive to minute details' (LeCun, Bengio and Hinton 2015, p. 438). The ability of Deep Learning (DL) algorithms to solve problems once thought impossible has caused a large amount of corporate interest, contrasting with the primarily academic interest when perceptrons were discovered. DL has even made its way into consumer products. Many modern phones have facial recognition software built in, and Artificial Intelligence (AI) voice assistants are becoming more common in homes. DL is also becoming more accessible, with the release of Google's 'Tensorflow' package for DL (Abadi *et al.* 2016), which is now available in 'Keras', a user friendly package for Python (Chollet *et al.* 2015) and R (Allaire and Chollet

2018). Given the amount of commercial success that DL has seen, another AI winter seems unlikely.

3 Gaussian processes

One proposed way to better understand DL is by using Gaussian Processes (GPs) to approximate the output of a DL algorithm.

In a similar way to how a univariate normal distribution with a mean and variance can be generalised to a multivariate normal distribution with a mean vector and a normal vector, a GP is the limit of extending a multivariate normal distribution to infinite dimensions, with a mean function and covariance function (sometimes also called a kernel in ML environments) which depends on the distance between two points.

A random vector $X \in \mathbb{R}^n$ is distributed with a multivariate normal distribution in n dimensions with mean vector $\boldsymbol{\mu} \in \mathbb{R}^n$ and covariance matrix $K \in \mathbb{R}^{n \times n}$ if $X \sim \mathcal{N}(\boldsymbol{\mu}, K)$. In a similar way, Y is a GP with a mean function $\mu(\,\cdot\,)$ and a covariance function $k(\,\cdot\,,\,\cdot\,)$ where $k(x_1, x_2) = k(|x_1 - x_2|)$ if $Y \sim \mathcal{GP}(\mu, k)$. Because a GP is an extension of a multivariate normal distribution, any finite subset of points from a GP has a multivariate normal distribution (Williams and Rasmussen 1996, p. 515).

3.1 History of Gaussian processes

GPs have been independently rediscovered many times in the 20th century in the contexts of time series, physics, mining, meteorology, computer experiments and machine learning (MacKay 1997, p. 2).

3.1.1 Time series

The origins of GPs can be traced at least as far back as WWII, and the work of Wold (1938) and Wiener (1949) on time series. They developed ARMA models and spline smoothing, which correspond to one dimensional GPs with specific choices of covariance functions.

3.1.2 Krige

In the 1950s, the problem of mine valuation — the decision of where to establish a gold mine — was often based on the sample mean and sample standard deviation of repeated biased sampling of the ore grade in an area. However, this simple statistical method could not accurately predict the volume of ore in an area and did not capture the variability of the ore density (Cressie 1990, p. 244; Krige 1951, p. 119). Many datapoints were needed to reduce the estimate of the ore density's variance to an acceptable level, and so often an insufficient number of samples were taken or money was wasted on taking too many samples. South African mining engineer and geostatistician Danie Krige[2] proposed some improvements to these methods, one of which was to take into account the distance of the samples from each other, which greatly reduced the number of samples needed (Krige 1951).

Despite the advantages of his ideas over the simpler techniques, his methods were still flawed. When estimating a value at a specific location $\boldsymbol{x}$, Krige's method weighted the datapoints by their distance from $\boldsymbol{x}$. However, he weighted all points at a distance greater than some threshold with zero weight and all other points with an equal weight, rather than weighting the datapoints proportionally to their distance from the point being estimated (Cressie 1990, p. 245). Krige later also experimented with including more points in his calculations, but did not put a smaller weight on more distant points, and so concluded that 'the advantage gained is of no practical significance' (Krige 1962, p. 362). Krige also used ordinary least squares regression where generalised least squares regression is the optimum for correlated squared errors, although for the large datasets Krige worked with, the inaccuracy caused was minimal.

3.1.3 Matheron

Krige did not see the value in weighting individual datapoints, saying 'any sample value cannot be regarded as having a so-called "area or distance of influence," in the generally accepted sense' (Krige 1951, p. 125). In the 1960s, French mathematician and geostatistician Georges Matheron formalised Krige's work and greatly expanded on it, coining the term 'krigeage'

[2] Afrikaans pronunciation: [dɑniː ˈkriχə]

(later anglicised to 'kriging') (Matheron 1960) and founding the field of geostatistics.

Matheron's way of formalising the 'area of influence' that Krige had dismissed was to create a function called the 'variogram', which represents the dissimilarity between two points and should increase as the distance between the points increases. '[The variogram] gives a precise content to the traditional concept of the influence zone of a sample' (Matheron 1963, p. 1250). He defined the variogram $2\gamma(\,\cdot\,,\,\cdot\,)$ as the variance of the difference between two points:

$$2\gamma(\boldsymbol{x}_i, \boldsymbol{x}_j) = \operatorname{Var}\left(y(\boldsymbol{x}_i) - y(\boldsymbol{x}_j)\right).$$

The semivariogram $\gamma(\,\cdot\,,\,\cdot\,)$ is half of the variogram and helps define how similarity behaves as distance between points changes. The 'nugget effect' is so named to reflect the fact that if gold has been found at a specific location, the probability of finding gold within a very small distance is also 1, i.e. still inside the gold nugget. For data points, the nugget represents a measure of the variability between points at zero distance from each other. Any variation at a distance less than the smallest sampling distance will be included in the nugget. The 'range' represents the maximum distance at which similarity stops decreasing, and the 'sill' represents the level of dissimilarity between points at a distance greater than the range.

Fitting a curve to a semivariogram plot such as Figure 5 can be used to model the covariance function (Diggle, Tawn and Moyeed 2002, p. 305) as the covariance function $k(\,\cdot\,,\,\cdot\,)$ is related to the semivariogram $\gamma(\,\cdot\,,\,\cdot\,)$ by:

$$\gamma(\boldsymbol{x}_i, \boldsymbol{x}_j) = \text{sill} - k(\boldsymbol{x}_i, \boldsymbol{x}_j).$$

As well as Krige, Matheron also was influenced by Gandin (1963, translated into English in 1966). This work was from the perspective of meteorology, Gandin's focus was more on the covariance function, whereas Matheron's geostatistics perspective meant he focused more on the variogram and semivariogram.

In 1986, Philip and Watson criticised Matheron's work and the field of Matheronian geostatistics from the perspective of geologists. They claimed the assumption of stationarity for ore density data was unrealistic (Philip and Watson 1986, pp. 86–87) and that least squares regression was inappropriate

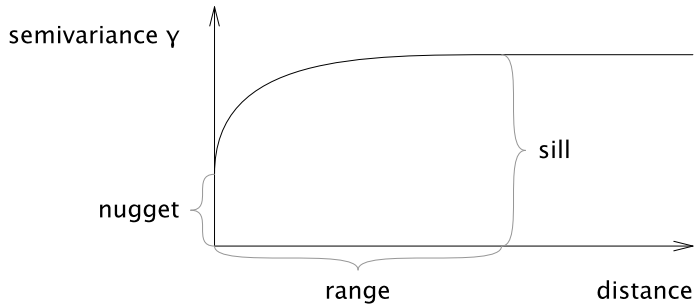


Figure 5: An example theoretical semivariogram plot showing the nugget, range and sill.

because of its assumption of normality (Philip and Watson 1986, pp. 101–102) and the disproportionate effect of outliers on the estimates (Philip and Watson 1986, p. 102).

3.2 Computer experiments

Another field in which GPs have been heavily used is in computer experiments, where the techniques were generalised to both more contexts and higher dimensions than two and three dimensional geostatistics. Instead of expensive physical experiments, large models are used to model physical phenomena such as circuit design, nuclear fusion, plant ecology, and thermal energy storage (Sacks *et al.* 1989, p. 409), usually involving large sets of differential equations. In cases such as weather, physical experimentation is impossible, so the only option is to solve these models by a complex computer program, which can be very computationally expensive, taking hours or days to complete a single run. It is therefore infeasible to complete multiple runs of the model with the same inputs for use in Monte Carlo simulations to calculate uncertainty like one would with a physical experiment. Additionally, the models are often deterministic, so even if multiple runs can be performed, the uncertainty in the estimate cannot be measured. One method to overcome these limitations is to run the computer model with a variety of input parameters and treat the outcomes as realisations of a GP model (Bachoc

et al. 2017, p. 163) This allows for interpolation of the model's outputs and quantification of the associated uncertainty of the computer model.

3.3 Modern trends

Modern trends in the field of GPs include GPs for big data, where the number of datapoints can be many millions or billions (Hensman, Fusi and Lawrence 2013). This means that traditional methods which involve inverting an $n \times n$ matrix cannot be used.

Deep GPs (or hierarchical GPs) are analogous to deep neural networks (see Section 2.6), and involve chaining the output of one GP into another to stronger model nonlinearity in the data (Damianou and Lawrence 2013).

4 Opening the black box

If deep neural networks are simply the combination of many simple mathematical operations in a seemingly random way, there may be a way to mathematically or statistically disentangle the meaning from the information in the network. Research carried out on 'inverting' a deep neural network that was trained on images in an attempt to reconstruct the original inputs found that each successive layer abstracts the data, but isolating the specific neurons responsible for specific objects or ideas was more difficult (Mahendran and Vedaldi 2014). Sensitivity analysis has also been used to evaluate the importance of each input to an ANN in the final output, and the results visualised as a decision tree where each branch represents a range of input values (Cortez and Embrechts 2013, p. 13).

There appears to be no literature on using GPs as a way to better understand how deep learning algorithms work or as a way to understand their structure, although there has been a fierce debate over the advantages of both ANNs and GPs since at least the 1990s. Rasmussen (1997, pp. 65–66) gave evidence that GPs are a valid replacement for neural networks in nonlinear problems with fewer than 1000 datapoints, although due to advances in processing power, ANN algorithms and GP techniques, this recommendation will have changed. MacKay (1997, p. 25) gave an example of GPs being used for binary classification, in a similar way to ANNs. When Gaussian Process Regression (GPR) is compared to ML methods, the main advant-

ages mentioned are usually faster convergence and confidence or credible intervals for the predictions, which ML cannot do (Herbrich, Lawrence and Seeger 2003).

5 Conclusion

Deep learning is fast becoming a ubiquitous tool in many aspects of our lives – sometimes clearly visible, as with driverless cars, but in some cases more discreetly, such as the use of ML algorithms in American courts. As deep neural networks were designed to mimic a biological brain, a phenomenon that is still not well understood, they have a 'black box' property that makes their reasoning is impossible to understand. GPs may offer a way to look inside this black box, as they offer a similar flexibility and wide range of uses, and are much more easily interpreted by humans. So far, much of the work that has been done involving GPs and ML has been comparative, rather than using one to model the other.

Bibliography

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. *et al.* (2016). 'TensorFlow: A system for large-scale machine learning'. 16, pp. 265–283. arXiv: `http://arxiv.org/abs/1605.08695v2 [cs.DC]`.

Allaire, J. J. and Chollet, F. (2018). *keras: R Interface to 'Keras'*. R package version 2.2.0. URL: `https://CRAN.R-project.org/package=keras`.

Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Hasan, M., Esesn, B. C. V., Awwal, A. A. S. and Asari, V. K. (2018). 'The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches'. *CoRR* abs/1803.01164. arXiv: `1803.01164`. URL: `http://arxiv.org/abs/1803.01164`.

Bachoc, F., Contal, E., Maatouk, H. and Rullière, D. (2017). 'Gaussian processes for computer experiments'. *ESAIM: Proceedings and Surveys* 60. Ed. by J.-F. Coeurjolly and A. Leclercq-Samson, pp. 163–179. DOI: `10.1051/proc/201760163`.

Bennett, J., Lanning, S. *et al.* (2007). 'The Netflix Prize'. *Proceedings of KDD cup and workshop*. Vol. 2007. New York, NY, USA, p. 35.

Block, H. D. (1970). 'A review of "Perceptrons: An Introduction to Computational Geometry"'. *Information and Control* 17.5, pp. 501–522. DOI: 10.1016/s0019-9958(70)90409-2.

Chao, X., Kou, G., Li, T. and Peng, Y. (2018). 'Jie Ke versus AlphaGo: A ranking approach using decision making method for large-scale data with incomplete information'. *European Journal of Operational Research* 265.1, pp. 239–247. DOI: 10.1016/j.ejor.2017.07.030.

Chollet, F. *et al.* (2015). *Keras*. https://keras.io.

Christin, A., Rosenblat, A. and Boyd, D. (2015). 'Courts and predictive algorithms'. *Data & CivilRight*.

Cortez, P. and Embrechts, M. J. (2013). 'Using sensitivity analysis and visualization techniques to open black box data mining models'. *Information Sciences* 225, pp. 1–17. DOI: 10.1016/j.ins.2012.10.039.

Cressie, N. (1990). 'The Origins of Kriging'. *Mathematical Geology* 22.3, pp. 239–252. DOI: 10.1007/bf00889887.

Damianou, A. and Lawrence, N. (2013). 'Deep gaussian processes'. *Artificial Intelligence and Statistics*, pp. 207–215.

Diggle, P. J., Tawn, J. A. and Moyeed, R. A. (2002). 'Model-based geostatistics'. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 47.3, pp. 299–350. DOI: 10.1111/1467-9876.00113.

Gandin, L. S. (1963). *Объективный анализ метеорологических полей (Objective analysis of meteorological fields)*. Vol. 287. Гидрометеоиздат Ленинград (Gidrometeoizdat).

— (1966). 'Objective analysis of meteorological fields'. Trans. by M. tirgume ha-mada' ha Yiśre'eli. *Quarterly Journal of the Royal Meteorological Society* 92.393, pp. 447–447. DOI: 10.1002/qj.49709239320.

Gerla, M., Lee, E., Pau, G. and Lee, U. (2014). 'Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds'. *2014 IEEE World Forum on Internet of Things (WF-IoT)*. IEEE, pp. 241–246. DOI: 10.1109/WF-IoT.2014.6803166.

Grey, C. G. P. (2017). *How Machines Learn*. URL: https://www.youtube.com/watch?v=R9OHn5ZF4Uo.

Gupta, M. M. and Qi, J. (1991). 'On fuzzy neuron models'. *IJCNN-91-Seattle International Joint Conference on Neural Networks*. Vol. 2. IEEE. IEEE, pp. 431–436. DOI: 10.1109/ijcnn.1991.155371.

Hensman, J., Fusi, N. and Lawrence, N. D. (2013). 'Gaussian Processes for Big Data'. arXiv: http://arxiv.org/abs/1309.6835v1 [cs.LG].

Herbrich, R., Lawrence, N. D. and Seeger, M. (2003). 'Fast sparse Gaussian process methods: The informative vector machine'. *Advances in neural information processing systems*, pp. 625–632.

Krige, D. G. (1951). 'A statistical approach to some basic mine valuation problems on the Witwatersrand'. *Journal of the Southern African Institute of Mining and Metallurgy* 52.6, pp. 119–139.

— (1962). 'Effective pay limits for selective mining'. English. *Journal of the Southern African Institute of Mining and Metallurgy* 62.6, pp. 345–363. ISSN: 0038-223X. URL: https://journals.co.za/content/saimm/62/6/AJA0038223X_3698.

LeCun, Y., Bengio, Y. and Hinton, G. (2015). 'Deep learning'. *Nature* 521.7553, pp. 436–444. DOI: 10.1038/nature14539.

Li, T., Katz, R. H. and Culler, D. E. (2018). 'ConNect: Exploring Augmented Reality Service using Image Localization and Neural Network Object Detection'.

Maass, W. (1997). 'Networks of spiking neurons: The third generation of neural network models'. *Neural Networks* 10.9, pp. 1659–1671. DOI: 10.1016/s0893-6080(97)00011-7.

MacKay, D. J. C. (1997). 'Gaussian processes: a replacement for supervised neural networks?'

Mahendran, A. and Vedaldi, A. (2014). 'Understanding Deep Image Representations by Inverting Them'. arXiv: http://arxiv.org/abs/1412.0035v1 [cs.CV].

Matheron, G. (1960). 'Krigeage dun Panneau Rectangulaire par sa Périphérie (Kriging a rectangular panel via its perimeter)'. *Note géostatistique* 28.

— (1963). 'Principles of geostatistics'. *Economic Geology* 58.8, pp. 1246–1266. DOI: 10.2113/gsecongeo.58.8.1246.

Minsky, M. and Papert, S. A. (1987). *Perceptrons: An Introduction to Computational Geometry, Expanded Edition*. The MIT Press. ISBN: 0-262-63111-3.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press Ltd. 1104 pp. ISBN: 0262018020. URL: https://www.ebook.de/de/product/19071158/kevin_p_murphy_machine_learning.html.

Olazaran, M. (1996). 'A Sociological Study of the Official History of the Perceptrons Controversy'. *Social Studies of Science* 26.3, pp. 611–659. ISSN: 0306-3127. URL: http://www.jstor.org/stable/285702.

Philip, G. M. and Watson, D. F. (1986). 'Matheronian Geostatistics—Quo Vadis?' *Mathematical Geology* 18.1, pp. 93–117. DOI: 10.1007/bf00897657.

Ramachandran, P., Zoph, B. and Le, Q. V. (2017). 'Searching for Activation Functions'. arXiv: http://arxiv.org/abs/1710.05941v2 [cs.NE].

Rasmussen, C. E. (1997). *Evaluation of Gaussian processes and other methods for non-linear regression*. University of Toronto.

Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton*. Cornell Aeronautical Laboratory.

— (1958). 'The perceptron: A probabilistic model for information storage and organization in the brain.' *Psychological Review* 65.6, pp. 386–408. DOI: 10.1037/h0042519.

Rumelhart, D. E., Hinton, G. E. and McClelland, J. L. (1986). 'A general framework for parallel distributed processing'. *Parallel distributed processing: Explorations in the microstructure of cognition* 1.45-76, p. 26.

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986). 'Learning representations by back-propagating errors'. *Nature* 323.6088, p. 533.

Sacks, J., Welch, W. J., Mitchell, T. J. and Wynn, H. P. (1989). 'Design and Analysis of Computer Experiments'. *Statistical Science* 4.4, pp. 409–423. ISSN: 0883-4237. URL: http://www.jstor.org/stable/2245858.

Wiener, N. (1949). *Extrapolation, Interpolation, and Smoothing of Stationary Time Series: With Engineering Applications*. The MIT Press. ISBN: 9780262230025.

Williams, C. K. I. and Rasmussen, C. E. (1996). 'Gaussian Processes for Regression'. *Advances in neural information processing systems*, pp. 514–520.

Wold, H. (1938). 'A study in the analysis of stationary time series'. PhD thesis. Almqvist & Wiksell.