University of Exeter

Can statistics help us to understand deep learning?

Johannes Smit

May 2019

Abstract

To-do: *All theses/dissertations must include an abstract of approximately 300 words bound in with each copy and placed so as to follow the title page.*

Acknowledgements

To-do: *Hi mum.*

Contents

Abstract **i**

Acknowledgements **ii**

1 Introduction **1**

1.1 What is machine learning? 1

2 Machine Learning **3**

2.1 Neural Networks 3

2.1.1 The Neuron 3

2.1.2 Neural network layers and Backpropagation 6

2.2 Deep Learning 7

2.3 Modern techniques/tools 8

2.4 Example 8

2.4.1 Training a neural network 8

2.4.2 Ordering of the datapoints 9

3 Opening the Black Box **12**

3.1 Previous papers on this subject 12

3.2 Regression 12

3.2.1 Linear regression 12

3.2.2 Stepwise regression 13

3.2.3 LASSO 13

3.3 Gaussian processes 14

3.3.1 Using Gaussian process regression 15

4 Conclusion 16
4.1 How well have each of the attempts worked? 16
4.2 What could be improved? 16
4.3 How useful would more research on this topic be? 16
4.4 What should future research on this topic focus on? 16

A Code 17

Bibliography 18

Chapter 1

Introduction

1.1 What is machine learning?

To-do: *Machine learning is very complex to human understanding*
Machine learning is widely used in many high stakes scenarios — give examples
It is important to be able to look inside the 'black box' of machine learning — explain how this would be useful in each of the given examples
We can use statistical methods to try and recover the information in a ML algorithm — give some idea how

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), 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 train a model. One of these 'machine learning' algorithms is the artificial neural network, which takes inspiration from the neural structure of biological brains. Artificial neural networks consist of units called 'neurons', which individually perform a simple nonlinear operation, but when connected, 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).

As these algorithms are placed in more real world scenarios and in control of important infrastructure and even human lives, the need to understand what is happening inside the 'black box' becomes more important. Autonomous vehicles are becoming more common, and 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 being trialled 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 artificial mind.

Chapter 2

Machine Learning

So much data is collected that in many cases, hand building a model to analyse, find patterns in and draw conclusions from the data is unfeasible, so we use computers to analyse it in a process called Machine Learning (ML) (Murphy 2012, p. 1). Computers are able to perform predictive tasks by combining information in nonlinear ways, but can become much 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 Neural Networks

2.1.1 The Neuron

The Perceptron

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.

Artificial Neurons

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 2.1, 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. The bias controls how large the weighted sum needs to be to

'activate' the neuron, so this is called the pre-activation (v).

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

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. The activation function must be differentiable if the gradient descent method is used (see Section 2.1.2) as gradient descent relies on calculating the gradient of the error.

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

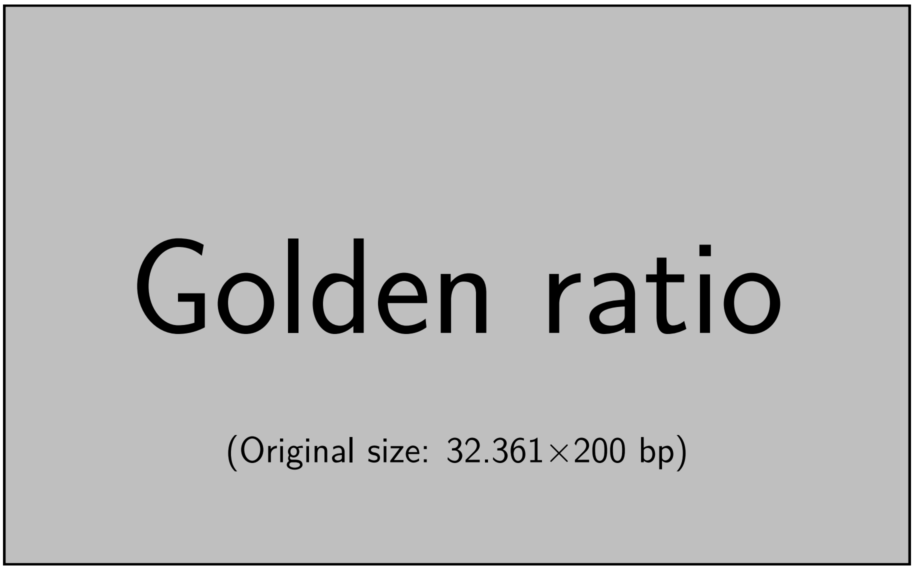

Figure 2.1: A diagram of an example artificial neuron.

Activation functions

Early examples of artificial neurons were built in hardware with the output being a light that was either on or off. This is equivalent to the 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 higher dimensions) are used. More recently, a commonly used activation function for deep learning (see Section 2.2) is the Rectified Linear Unit (ReLU) function (Ramachandran, Zoph and Le 2017), which is defined as the positive part of its input $\mathrm{ReLU}(x) = \max(0, x)$. The ReLU function was designed to be analogous to how a biological neuron can be either inactive or activated, although why it works as well as it does is not well understood. [Note: *Rewrite that last sentence to not include 'works as well as it does'.*] If the activation function is the identity function, then optimising a neuron is equivalent to performing linear regression.

2.1.2 Neural network layers and Backpropagation

> **To-do:** *Neurons are connected in layers.*
> *Backpropagation is used to optimise the weights.*
> *Batch backpropagation of all the datapoints takes too long.*
> *Online BP is faster but takes more steps.*
> *Stochastic minibatches can be used to speed up calculations.*

The weights in a neural network are optimised using the backpropagation algorithm.

> **To-do:** *[Note: Fix my explanation of the backpropagation algorithm.]*
> *For a neural network with N weights*
>
> 1. *Randomly initialise all the weights $\boldsymbol{w} = w_1, \ldots, w_N$.*
> 2. *Predict the outputs $\boldsymbol{y}^{pred} = y_1^{pred}, \ldots, y_n^{pred}$ using the training data $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$.*

3. *Calculate the error vector* $\boldsymbol{\epsilon} = \boldsymbol{y}^{pred} - \boldsymbol{y}^{target}$. *The error only depends on the output of .*

4. *Calculate the gradient of the error* $\boldsymbol{\Delta} = \frac{\partial \boldsymbol{\epsilon}}{\partial \boldsymbol{w}}$. $\boldsymbol{\Delta}$ *represents the direction in* N*-dimensional space...*

5. *Update the weights* $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta\boldsymbol{\Delta}$.

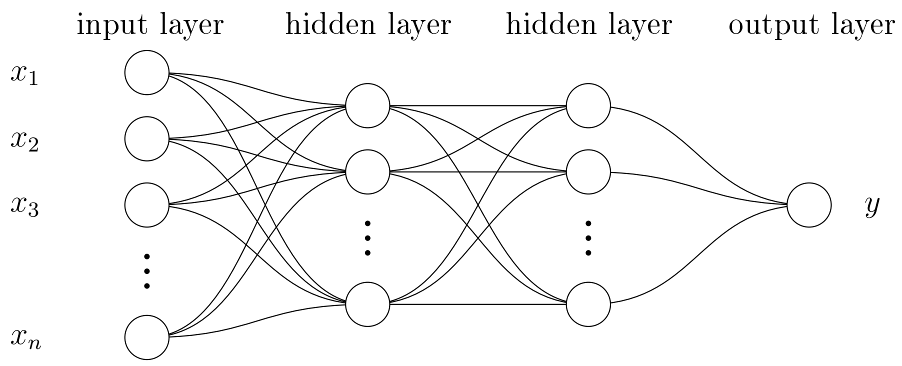

Figure 2.2: An example of the structure of a neural network.

Estimating the parameters of an Artificial Neural Network (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).

2.2 Deep Learning

To-do: *DL is just ML with more layers*

To-do: *Rewrite this to more slowly introduce deep learning and be more generally about machine learning than ANNs*

As computing power has increased exponentially following Moore's law, machine learning has become a more viable and effective method of prediction. In the 21st century, success has been found in increasing the number of layers in a neural network or other machine learning method rather than the complexity of each layer. 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).

2.3 Modern techniques/tools

Deep Learning (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, for example with the release of Google's 'Tensorflow' package for DL (Abadi *et al.* 2016), which is now available as 'Keras', a user friendly package for Python (Chollet *et al.* 2015) and R (Allaire and Chollet 2018).

2.4 Example

To demonstrate the possibility of using statistical methods to understand the process of deep learning, we use a simple function $f(x) = x + 5\sin(x) + \epsilon$, where ϵ is iid normal noise $\epsilon \sim \mathcal{N}(0, 0.1)$. 256 evenly spread datapoints were taken from this function, seen in Figure 2.3.

2.4.1 Training a neural network

This function was learnt by a neural network with 8 layers, each with 10 neurons and the $\tanh(\cdot)$ activation function, except for the final layer which used

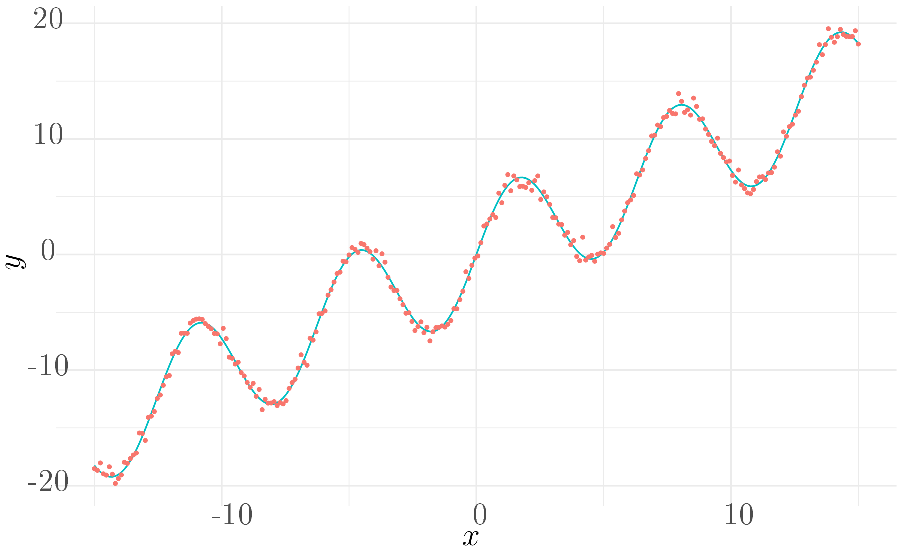

Figure 2.3: The true function $f(x) = x + 5\sin(x)$ (blue) and the training data (red).

a linear activation function. The result of this learning is seen in Figure 2.4.

As ANNs get deeper, they reach the ‘vanishing gradient problem’, where the gradient changes to each layer get smaller and smaller as the backpropagation algorithm gets closer to the input layers. An ANN with too many parameters/too wide will just ‘remember’ the training data, i.e. overfit the data and not find any meaningful patterns in the data.

To-do: *Add some more information on how to choose the right size ANN.*

2.4.2 Ordering of the datapoints

Due to the stochastic batch gradient descent used to train the model, if the order of the datapoints is not randomised, then the optimisation algorithm is significantly more likely to get stuck in a local minimum. An example of this is seen in Figure 2.5, where the same neural network has been trained on the example data which has been shuffled, reversed, randomised and separated.

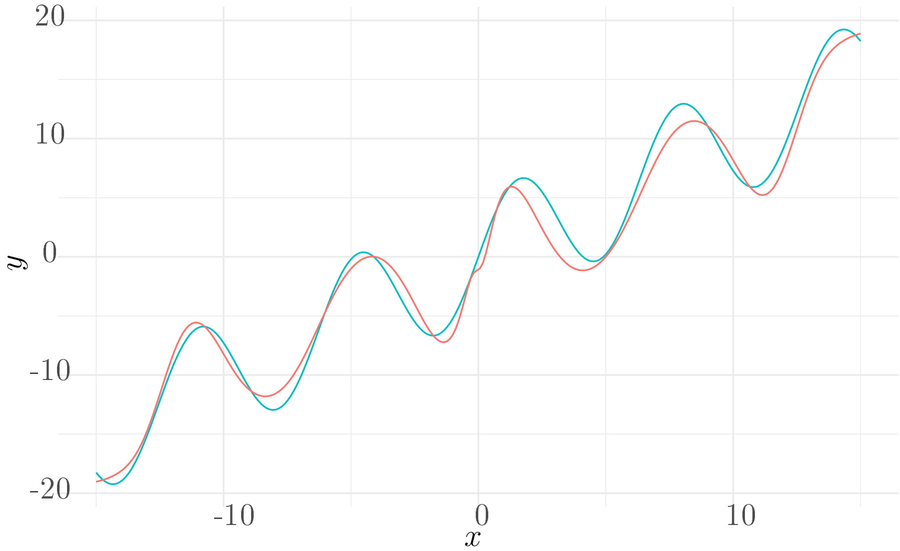

Figure 2.4: The true values (blue) and the ANN's predicted values (red).

To-do: *Expand on what these mean.*

Bengio *et al.* (2009) developed a technique called 'curriculum learning' which uses this fact to improve training by starting with easier training examples and slowly working towards more difficult training examples.

To-do: *Expand on this topic.*

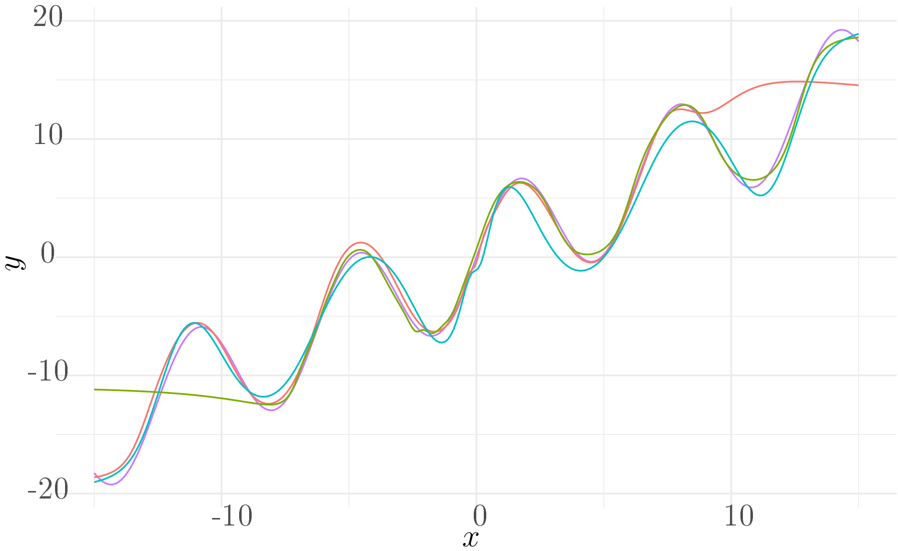

Figure 2.5: The output of the same neural network trained on the same data but ordered differently.

Chapter 3

Opening the Black Box

3.1 Previous papers on this subject

To-do: *Add some previous research here.*

3.2 Regression

One method of opening the black box of machine learning is to use multiple regression.

3.2.1 Linear regression

In this case, x, x^2, $\sin(x)$, $\sin(2x)$, $\sin(x/2)$, $\cos(x)$, $\cos(2x)$ and $\cos(x/2)$ were used. This produced a complex model with many coefficients close to zero.

To-do: *Expand on why I chose these basis functions/vectors.*

To-do: *This is slightly cheating, but is feasible and a common technique. Find a source for this.*

Table 3.1: The results of the multiple regression model.

term	estimate	standard error	test statistic	p-value
x	0.939	0.007	143.719	0.000
x^2	0.003	0.001	3.444	0.001
$\sin(x)$	4.680	0.080	58.640	0.000
$\sin(2x)$	0.147	0.079	1.856	0.065
$\sin(x/2)$	-0.412	0.082	-5.017	0.000
$\cos(x)$	-0.080	0.081	-0.983	0.327
$\cos(2x)$	-0.026	0.080	-0.318	0.751
$\cos(x/2)$	-0.110	0.088	-1.253	0.211

Table 3.2: The results of the model reduced using stepwise AIC.

term	estimate	standard error	test statistic	p-value
x	0.939	0.007	143.758	0.000
x^2	0.003	0.001	3.184	0.002
$\sin(x)$	4.680	0.080	58.656	0.000
$\sin(2x)$	0.147	0.079	1.857	0.064
$\sin(x/2)$	-0.412	0.082	-5.018	0.000

3.2.2 Stepwise regression

It is then possible to use stepwise regression to reduce the number of parameters in this linear model. In this case, the variables were selected using the Akaike information criterion (AIC).

While the relevant estimators x and $\sin(x)$ were identified and their coefficients fairly accurately estimated, a few other variables were also identified as significant. This method is only likely to work with a perfect or near perfect fit with no noise, which is unrealistic for real applications.

3.2.3 LASSO

Alternatively, we can use the Least Absolute Shrinkage and Selection Operator (LASSO) method to select the significant variables from the full model. The LASSO method constrains the sum of the absolute values of the model parameters, regularising the least influential parameters to zero. We vary

Table 3.3: The coefficients of LASSO CV.

term	estimate
x	0.927
x^2	0.001
$\sin(x)$	4.490
$\sin(x/2)$	-0.207

the hyperparameter λ to change how regularised the coefficients are, as seen in Figure 3.1. We then use k-fold cross validation to find the optimal hyperparameter λ.

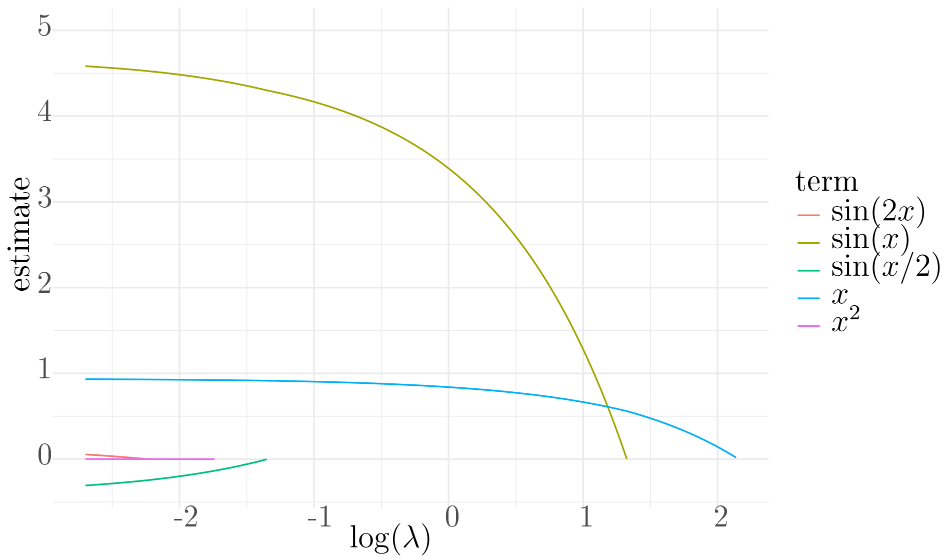

Figure 3.1: Changing the hyperparameter λ.

3.3 Gaussian processes

We can use Gaussian Processes (GPs) to try and model the output of the ANN. GPs are an extension of the multivariate normal distribution to an infinite dimensional process with a mean function and covariance function instead of a mean vector and covariance matrix.

3.3.1 Using Gaussian process regression

Because GPs are very flexible, applying a GP to the output of the ANN is likely to result in a close fit.

To-do: *The limit of an ANN is a GP.*

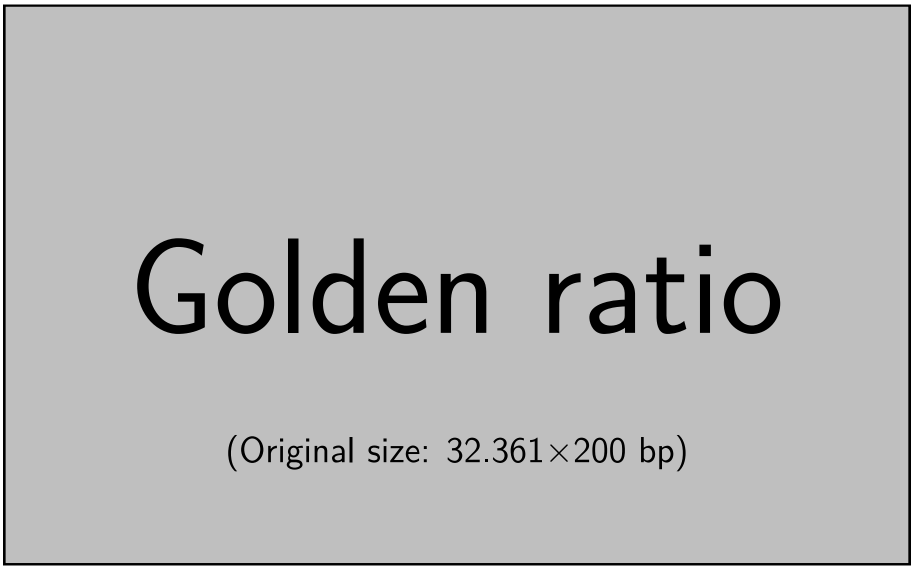

Figure 3.2: The fit of the GP.

Chapter 4

Conclusion

4.1 How well have each of the attempts worked?

> **To-do:** *The stepwise regression was not as good as the LASSO technique, which worked quite well.*

4.2 What could be improved?

4.3 How useful would more research on this topic be?

4.4 What should future research on this topic focus on?

Appendix A

Code

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`.

Bengio, Y., Louradour, J., Collobert, R. and Weston, J. (2009). 'Curriculum learning'. *Proceedings of the 26th annual international conference on machine learning.* ACM, pp. 41–48.

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.*

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`.

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’.

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`.

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

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`.