LU-TP 22-28
June 2022

Combined Regularisation Techniques for Artificial Neural Networks

Joseph A. Binns

Department of Astronomy and Theoretical Physics, Lund University

Bachelor thesis
Supervised by Patrik Edén

LUND
UNIVERSITY

Abstract

Artificial neural networks are prone to overfitting – the process of learning details specific to a particular training data set. Success in preventing overfitting through combining the L_2 and dropout regularisation techniques has led to the combination's recent popularity. However, with the introduction of each additional regularisation technique to an artificial neural network, there comes new hyperparameters which must be tuned in an increasingly complex and computationally expensive manner. Motivated by L_2's action as a Gaussian prior on the loss function, we hypothesise an analytic relation for an optimal L_2 strength's dependence on the number of patterns. Conducted on an artificial neural network composed of a single hidden layer, this systematic study tests the hypothesis for optimal L_2 strength, and considers what interactions the additional involvement of dropout and early stopping may have on the relation. On an otherwise static problem and network calibration, the results of this thesis suggested the success of the hypothesis within a valid working region. The results are useful informants for the choice of L_2 strength, drop rate and early stopping usage, and gave promise that the predictor may find real world applications.

Popular Science Description

Artificial Intelligence's (AI's) potential for incredible state-of-the-art performance has not gone unnoticed; from medicine to military, the interest of all manner of fields has been peaked [1]. This has encouraged the rapid integration of AI into our everyday lives [2]. However, in the recent swarm of industrial excitement, whilst new applications have taken the limelight, rigour and understanding have began to lag behind. By shining light on a popular choice of mechanisms which assist in the training of AI, known as dropout, L_2 and early stopping, my study aimed to be a small step towards designing AI in a more informed and understood manner.

Artificial Neural Networks (ANNs) are a collection of computational architectures inspired by the brain; they are the current most realised form of AI. If an ANN is insufficient in size, it will lack the capacity to solve even the simplest of problems. However, if an ANN is too large, then that excessive capacity seldom lies dormant. Instead, in a process known as overfitting, the ANN tends to learn undesirable peculiarities in a data set, such as fuzzy noise. This, in turn, can result in an ANN that generalises poorly to new data – a tendency to perform insufficiently on previously unseen variations of the same underlying problem [3].

Driven by a desire to suppress overfitting, there have been a variety of developments of so-called regularisation techniques. L_2, dropout and early stopping are common such choices. In particular, L_2 and dropout have recently received praise and popularity for providing good results when applied in conjunction [4, 5]. Though regularisation techniques offer significant benefits – often being of practical necessity – their implementation does not come without its costs. Notably, both L_2 and dropout have associated values controlling their strengths, each of which must be exhaustively fine-tuned to the specific problem and chosen ANN architecture [6].

To guide in what can become a lengthy and troublesome process of trial-and-error, my study aimed to test a hypothesised predictor for optimal L_2 strength. The predictor proposed that optimal L_2 strength is proportional to the amount of available training data. The effects on optimal L_2 strength, of using L_2 in conjunction with both the dropout and early stopping regularisation techniques, were then observed.

The results, which suggest the predictor to be successful within a suitable region, have helped to improve understanding of the interactions between these combined regularisation techniques. There shows promise that the predictor may find real world usage from it's extrapolation to situations with many training patterns, which would otherwise rely upon a time-consuming hyperparameter search.

Contents

List of Acronyms

AI Artificial Intelligence

ANN Artificial Neural Network

MLP Multi-layer Perceptron

SGD Stochastic Gradient Descent

List of Figures

List of Tables

List of Algorithms

1

Introduction

After an initial period of training, Artificial Neural Networks (ANNs) are prone to overfitting – the process in which a network trains too exactly to a data set. Overfitting is problematic, as it often results in the learning of undesirable peculiarities in the training data, such as fuzzy noise. In turn, this can cause an ANN to generalise poorly, even on data sets belonging to the same statistical distribution. Regularisation techniques are attempts to improve generalisation by suppressing overfitting, without jeopardising performance [3].

L_2 regularisation, a common regularisation technique, acts to minimise complexity by penalising excessively large weights. However, there is a fine line to be drawn; too much L_2 regularisation, and large weights will be penalised so much that the model will tend to set all weights to zero. When designing an ANN, the L_2 regularisation strength λ must therefore be varied in order to determine it's optimal value [7].

Dropout, another regularisation technique, is the effective idea of omitting weights connected to random nodes during training. Dropout thus helps suppress the collaboration of nodes, which can otherwise allow for the accommodation of noise-induced outliers. The L_2 and dropout regularisation techniques are popular choices, distinguished for providing good results when applied in conjunction [4, 5].

Early stopping regularisation, the final technique of concern, halts training at the first sign of overfitting; this can potentially improve generalisation performance and reduce the required number of epochs. Whilst the technique can completely avoid overfitting during training, it's use is often criticised for lacking elegance. Namely, it has been argued that early stopping replaces one problem with another, as it introduces the risk of overfitting to the validation data set [3]. Regardless of it's shortcomings, early stopping is important to understand, as it can be employed unintentionally if other settings prevent training from converging.

With the introduction of each additional regularisation technique, there come new hyperparameters which must be tuned. As the list of hyperparameters is already extensive, the task of hyperparameter selection is complex and difficult. Despite the fact that it is both computationally expensive and time consuming, random hyperparameter selection through trial-and-error remains the gold-standard [6].

My study aimed to inform this often convoluted process by testing a hypothesised predictor for optimal L_2 strength. The hypothesis – motivated by L_2's action as a Gaussian prior on the loss function [8] – proposes that the optimal of λ is inversely proportional to the number of training patterns. The effects on optimal λ, of using L_2 in conjunction with dropout and early stopping, were then observed.

2

Theory

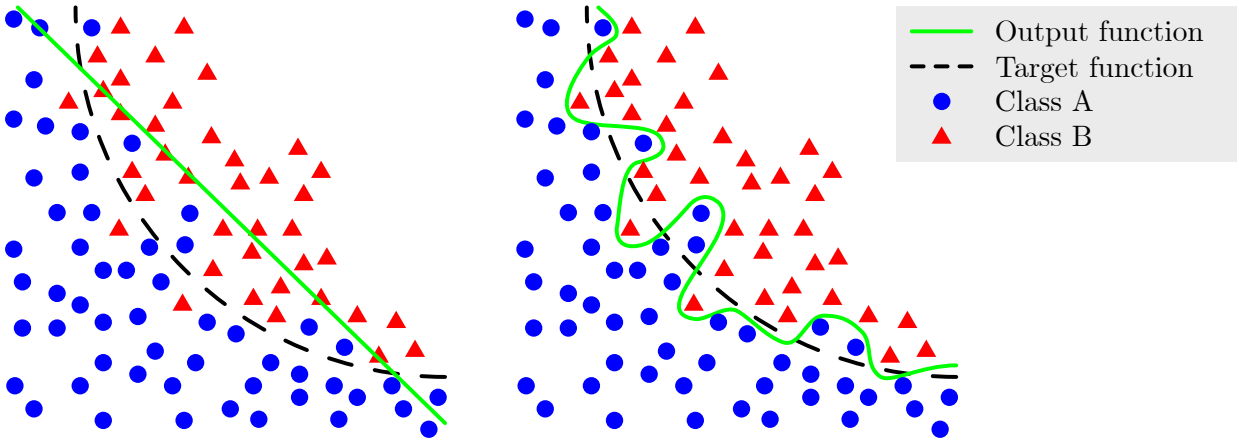
ANNs are a collection of computational models inspired by the functioning of the brain. The fundamental building block of ANNs is the node – a simple computational unit, akin to the biological neuron. Each node has input and output connections with associated weights, including a bias term which controls the node's critical input level. To calculate a node's output, a chosen output function is applied to the sum of the bias term and a weighted average over the node's inputs. Over a period of training, the weights are adjusted to best solve the given problem. Adjustments to the weights are driven by the minimisation of a loss function $E(\boldsymbol{\omega})$, in a process known as gradient descent [3]. A single node alone is only capable of solving linearly separable problems. However, when multiple nodes are arranged into elaborate architectures, such as the Multi-layer Perceptron (MLP) exemplified in figure 2.2a, ANNs have proved capable of solving increasingly difficult Turing-computable problems [9].

2.1 Bias-variance trade-off

Whilst the capacity to solve problems increases with larger and more elaborate networks, so does the risk that the network may over-train to a property specific to the training data set, such as noise. This relationship can be described by the bias-variance trade-off [10]. Bias and variance are defined as follows:

- **Bias**, not to be mistaken for the node-bias term, is a measure of how much the ANN's output function differs from the target function [3].
- **Variance**, σ^2, measures the output's sensitivity to the data set [3].

In keeping with these definitions, the golden standard of ANNs – for which the output function approaches target function – requires that bias and variance are simultaneously minimised. In order to obtain a low bias, the ANN must have a sufficient capacity, which is typically achieved by increasing the number of hidden nodes. However, too much capacity, and the ANN faces the risk of overfitting to the particular data set, resulting in a stark increase in variance. This trade-off between bias and variance is illustrated in figure 2.1, where it is shown that the cost of a low variance through underfitting is a high bias, and the cost of a low bias through overfitting is a high variance. This proves problematic in the search for an ANN that is both accurate and generalisable [10].



(a) Underfitting (b) Overfitting

Figure 2.1: Typical decision boundaries from heavily under-fitted and over-fitted MLPs

2.2 Regularisation techniques

Regularisation techniques are attempts to overcome the bias-variance trade-off. A common symptom of overfitting, as seen in figure 2.1b, is the accommodation of outliers through regions of high curvature. Since it is only the weights and not the node-biases that control the output curvature, the node-bias term is typically exempt from regularisation [3].

2.2.1 Weight Decay

Weight Decay, or L_2 regularisation, is a technique which modifies the loss function $E(\boldsymbol{\omega})$ – used during training to calculate the weight updates. The loss function is modified to include a term that penalises exceptionally large weights ω; without this term, it would allow for the accommodation of outliers, and hence the learning of noise specific to the training data set. The modified loss function $\tilde{E}(\boldsymbol{\omega})$ is described mathematically in equation 2.2.1 [7].

$$\tilde{E}(\boldsymbol{\omega}) = E(\boldsymbol{\omega}) + \lambda\Omega(\boldsymbol{\omega}) \tag{2.2.1}$$

Here, λ is the L_2 strength and $\boldsymbol{\omega}$ the weights of the ANN. A choice of $\lambda = 0$ is thus equivalent to the original loss function. The Weight Decay regularisation term $\Omega(\boldsymbol{\omega})$ is defined as a summed square over all the weights, bar node-biases:

$$\Omega(\boldsymbol{\omega}) = \frac{1}{2}\sum_i \omega_i^2 \tag{2.2.2}$$

2.2.1.1 Provisional hypothesis

From a desire to find an analytical relation for the dependence of optimal L_2 strength λ^* on the number of training patterns N from a data set $\mathcal{D}$, we proposed a provisional hypothesis:

$$\lambda^* \propto \frac{1}{N} \tag{2.2.3}$$

Derivation The hypothesis was motivated by the conventional maximum likelihood interpretation of the loss, $E(\boldsymbol{\omega}) = -\frac{1}{N} \ln p(\mathcal{D}|\boldsymbol{\omega})$, and a Bayesian interpretation of the regularised loss, $\tilde{E}(\boldsymbol{\omega}) = -\frac{1}{N} \ln p(\boldsymbol{\omega}|\mathcal{D})$. From Bayes' theorem [11], the hypothesis' derivation began from a search for a suitable constant of proportionality for the L_2 choice of prior:

$$\begin{aligned} p(\boldsymbol{\omega}|\mathcal{D}) &= \frac{p(\mathcal{D}|\boldsymbol{\omega})p(\boldsymbol{\omega})}{p(\mathcal{D})} \\ \iff \ln p(\boldsymbol{\omega}|\mathcal{D}) &= \ln p(\mathcal{D}|\boldsymbol{\omega}) + \ln p(\boldsymbol{\omega}) - \ln p(\mathcal{D}) \end{aligned}$$

After disregarding terms independent of $\boldsymbol{\omega}$, the substitution of $E(\boldsymbol{\omega})$ and $\tilde{E}(\boldsymbol{\omega})$ was facilitated by multiplication by a factor of $-\frac{1}{N}$. This gave a form equivalent to equation 2.2.1, hinting that $\lambda\Omega(\boldsymbol{\omega}) = -\frac{1}{N} \ln p(\boldsymbol{\omega})$:

$$\begin{aligned} -\frac{1}{N} \ln p(\boldsymbol{\omega}|\mathcal{D}) &= -\frac{1}{N} \ln p(\mathcal{D}|\boldsymbol{\omega}) - \frac{1}{N} \ln p(\boldsymbol{\omega}) + \text{constant} \\ \iff \tilde{E}(\boldsymbol{\omega}) &= E(\boldsymbol{\omega}) + \lambda\Omega(\boldsymbol{\omega}) \end{aligned}$$

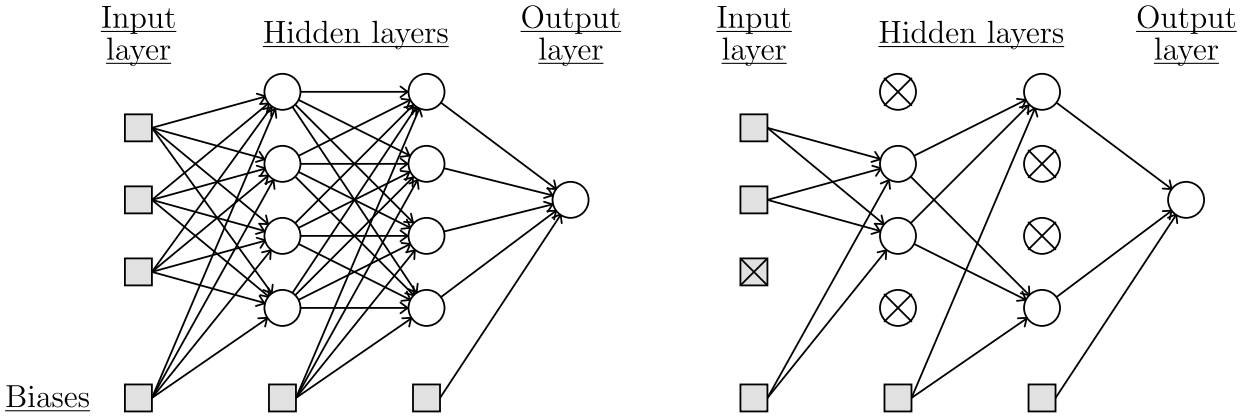
Hence, the L_2 choice of prior was hypothesised to have a constant of proportionality $\frac{1}{N}$.

2.2.2 Dropout

A procedure found to consistently improve generalisation is that of collecting ensembles of individual neural networks, from which an averaged output is then calculated and used. The technique favours a diverse collection of networks. Such a collection of accurate networks with low bias and high diversity is optimal, with an average output that should tend towards having a low bias and low variance – circumventing the bias-variance trade-off. However, exceptional performance does not come without its expense. Often in order to create a sufficient diversity of networks, the architecture of the ANN itself must be altered across the ensemble members. Consequently, substantial time can be consumed by the hyperparameter tuning and training of the diverse ANN's [4].

Dropout is a technique which was introduced to combine a vast amount of ANN architectures in an efficient manner. For each pattern, every node has a probability P of being "dropped". P is typically set per layer, where $P = 0$ is equivalent to no dropout. If a node is dropped, then all weights to and from the node are temporarily ignored from calculations, resulting in an altered weight update. Therefore, dropout effectively acts as

a low-cost ensemble across a variety of architectures, briefly performed over a single period of training. Because of this, dropout is able to prevent nodes from collaborating to fit outliers [3, 5].



(a) MLP (b) MLP after applying dropout

Figure 2.2: MLP without and with dropout

Dropout is typically applied to the input layer and hidden layers; common to other regularisation techniques, dropout is not applied to node-biases. Dropout provides maximum regularisation at $P = 0.5$, for which P is typically close to optimum for hidden units. Larger P is ill-advised, as connections receive less training, without benefiting from greater regularisation. For input nodes, a lesser drop rate of $P = 0.2$ is often considered more suitable, as it is advantageous to retain training data [4, 12].

2.2.3 Early stopping

After an ANN is initialised and prior to training, it is typical to have a high bias and low variance. Over the course of training, bias tends to decrease whilst variance tends to increase. Eventually, when validation loss E is minimised, the effect of the increasing variance begins to overcome that of the decreasing bias, as illustrated in figure 3.2 [10]. Early stopping prevents overfitting by halting training at this minimal validation loss. However, whilst overfitting to the training data set may be prevented, the observation of validation loss, which informs this process, introduces the risk of instead overfitting to the validation data set. The reliance on the validation data set additionally spoils the orthogonalisation of hyperparameter selection, which can prove especially problematic when used in conjunction with other regularisation techniques [13].

3

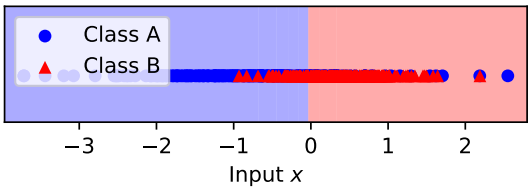
Methodology

Through a systematic study of combined regularisation techniques, I investigated optimal L_2 strength's dependence on the number of training patterns, for a variety of typical drop rates and instances of early stopping. To facilitate this investigation, a problem and model for the ANN were chosen. In order to limit the scope of the study, the ANN was chosen to be a MLP with a single hidden layer. The MLP was then trained using gradient descent. Stochastic Gradient Descent (SGD) was not used, as it can itself be considered a regularisation technique, which would risk adding confusion to the results. The synthetic data set and the MLP were generated in Python using our own code base [14].

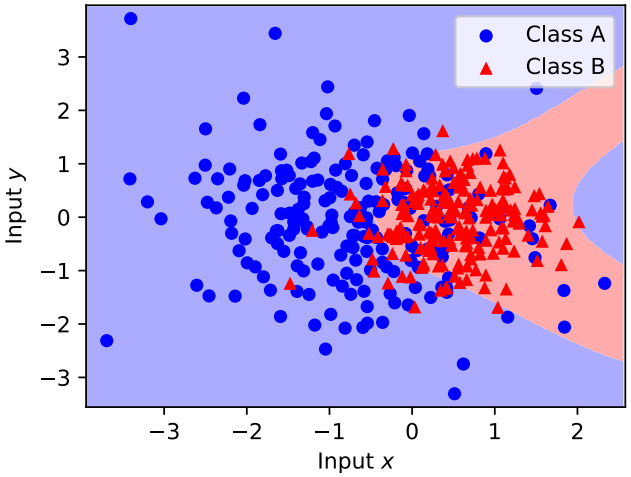
3.1 Data set

The chosen problem was a simple binary classification composed of two equally-distributed, overlapping Gaussian distributions, each with an assigned class [3]. To ensure that the optimal of λ would not favour an overwhelming suppression of weights, the target decision boundary would need to be non-linear. Consequently, the second distribution was chosen to have half the standard deviation of the first. The centers of the distributions were then displaced along a single axis by a distance equal to the larger of the standard deviations.

The number of dimensions of the synthetic data set was varied in order to get sufficient overfitting on a problem that wasn't overwhelmingly difficult. As the number of dimensions increased, the amount of overfitting increased whilst the problem difficulty decreased. This relationship can be explained by the decision boundary having more dimensions of freedom with which to accommodate outliers. This is demonstrated for the 1D and 2D versions of the data set in figure 3.1. Preliminary results from a selection of dimension-varied takes on the problem deemed the 10D case to be sufficiently responsive to regularisation.



(a) 1D data set



(b) 2D data set

Figure 3.1: The 1D and 2D versions of the data set, demonstrating a decision boundary with more dimensions of freedom to better avoid outliers

The 10D case of the problem was used consistently across all trials; variations were made only to the number of patterns, and the seed of the random number generator responsible for synthesising the data set. The number of patterns and the seed were respectively varied for investigative purposes and the approximation of errors. Since the stability of the results relied upon a suitably large validation data set, 4 times the number of validation patterns were used relative to training patterns N.

3.2 Random hyperparameter search

Once the data set was established, a broad tuning of the ANN's hyperparameters was performed. Since the provisional hypothesis involves a loss-minimising λ, validation loss was chosen as the primary performance measure. The constant hyperparameters shown in table 3.1a were tuned early-on whilst using the default hyperparameters in table 3.1b, and were subsequently left unchanged in all the presented investigations. To ensure that the network with dropout would not suffer from insufficient capacity, a generous number of hidden nodes was chosen. Due to a lesser P being favoured for inputs, to reduce the number of hyperparameters, dropout was not applied to the input layer. As such, the presented drop rate of P is only applicable for the hidden layer.

Constant Hyperparameters	
Activation function	tanh
Learning rate	0.1
Hidden nodes	20

(a) Hyperparameter values that were held constant

Default Hyperparameters	
Epochs	4000
L_2 strength, λ	0.0
Drop rate, P	0.0
Training patterns, N	10^3

(b) Hyperparameters take these default values, unless stated otherwise

Table 3.1: Constant and default hyperparameter values

To aid in the cumulative collection of results, random hyperparameter search was chosen over grid-search [6]. Using a random hyperparameter search, algorithm 3.1 describes the procedure for creating a heat map of λ against P for a fixed number of epochs, coloured by final validation loss E. Plots like this were used to help inform of L_2's interaction with dropout. To enhance illustrations, an irregular triangular grid was formed from the existing runs, upon which cubic interpolation was applied and clamped [15].

An informed investigation into how the optimal of λ varies with N was then conducted, testing the hypothesis. The procedure for generating a plot of optimal L_2 strength λ^* against N for a given drop rate and number of epochs is seen in algorithm 3.2. I chose to plot and calculate the standard errors of $\log\langle\lambda^*\rangle$ as opposed to $\langle\log(\lambda^*)\rangle$, since in the calculation of the mean in the latter, regardless of whether there is just a single $\lambda^* = 0$ or many, the resulting value would be 0; this could have proved misleading by risking reducing the range over which regressions could be made.

Algorithm 3.1 L_2 strength λ against drop rate P heat map

Generate training data set $\mathcal{D}_{training}$
Generate validation data set $\mathcal{D}_{validation}$
while there are fewer pairs of P and λ than desired **do**
    $P \leftarrow$ random uniformly-distributed number $\in [0, 1]$
    $\lambda \leftarrow$ random log uniformly-distributed number $\in [10^{-6}, 1]$
    Create the ANN
    Train the ANN using $\mathcal{D}_{training}$
    Validate the ANN using $\mathcal{D}_{validation}$
    $E \leftarrow$ final validation loss
    Store this pair of P and λ, with a reference to E
end while
Plot $\log(\lambda)$s against Ps as a heat map, colouring each point by its respective E

Algorithm 3.2 Optimal L_2 strength λ^* against number of training patterns N plot

Ns $\leftarrow$ array containing the number of patterns N to try
for each $N \in N$s **do**
  λ^*s $\leftarrow \emptyset$ ▷ For each $\mathcal{D}_{\text{validation}}$ tested, this set will contain a λ^*
  while there are fewer λ^* than desired **do**
    Generate training data set $\mathcal{D}_{\text{training}}$ from N patterns
    Generate validation data set $\mathcal{D}_{\text{validation}}$ from $4N$ patterns
    Es $\leftarrow \emptyset$ ▷ For each λ tested, this set will contain a E
    while there are fewer λ than desired **do**
      $\lambda \leftarrow$ random log uniformly-distributed number $\in [10^{-6}, 1]$
      Create the ANN
      Train the ANN using $\mathcal{D}_{\text{training}}$
      Validate the ANN using $\mathcal{D}_{\text{validation}}$
      $E \leftarrow$ final validation loss
      Insert E into Es
      Store this λ, with a reference to E
    end while
    $\lambda^* \leftarrow \lambda$ which references $\min(E\text{s})$
    Insert λ^* into λ^*s
  end while
  $\varepsilon \leftarrow$ the standard error of λ^*s
  Store this pair of $\log\langle\lambda^*\rangle$ and ε, with a reference to N
end for
Plot $\log\langle\lambda^*\rangle$s, with error-bars εs, against $\log(N)$s
In the region which visual inspection deems suitable, fit a linear regression to the plot

3.3 Early stopping

Without regularisation, for a variety of training patterns N, plots of validation loss E over epochs were created. As indicated in figure 3.2, minimum E occurred at greater epochs as N increased. To see what interactions early stopping may have with L_2 and dropout, the number of epochs was reduced from 4000 to 1000. From looking at the minimum validation losses, we know to expect that at 1000 epochs of training, early stopping will occur for approximately $N \geq 10^{2.5}$. Whilst early stopping won't typically occur for $N < 10^{2.5}$, we know to instead expect a reduced amount of overfitting.

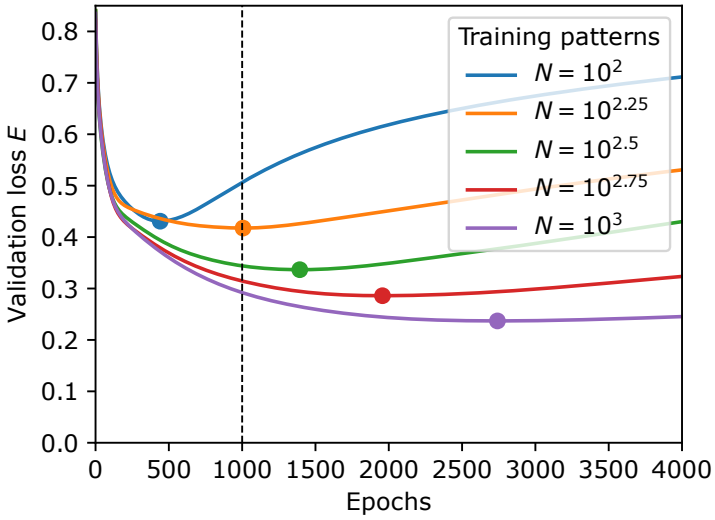


Figure 3.2: Typical validation loss over epoch plots for varying amounts of patterns, without L_2 or dropout

4

Results

The following statements are repeated here as a reminder to the reader:

1. $\mathbf{L}_2$ strength λ was applied equally across all layers.

2. **Dropout** was not applied to the input layer, such that the presented drop rate of P is only applicable to the hidden layer.

3. N refers to the number of training **patterns**, whilst the number of validation patterns was $4N$.

4.1 Regular epochs for substantial overfitting

Trained for 4000 epochs, figure 4.1 depicts heat maps of final validation loss E for runs varying in L_2 strength λ and drop rate P. The seeds responsible for the data set, weight initialisation and dropout were held constant for a given heat map. It is therefore important to note that, although the loss landscapes presented are quite typical, there is some susceptibility to variation.

The results, as shown in figure 4.1, suggest that the number of training patterns N both affects the range of observed final validation losses E, and the location and shape of the region over which E is minimal. In particular, it was seen that as N increased, E typically decreased, with the region over which E was minimal shifting to lesser L_2 strengths and drop rates. Highlighted on the color-bar is $E = 0.693 \approx \ln(2)$. For binary classification with equally distributed classes, this value corresponds to the optimal loss for an uninformed classifier – one which assumes the same output to every pattern. As this value of validation loss is consistently seen across large λ, L_2 is likely suppressing weights so much as to enforce a pattern-independent output. When the ANN was trained on as exceptionally few patterns as $N = 10$, a minimum loss of $E \approx 0.693$ in figure 4.1a suggested that the ANN held no predictive power.

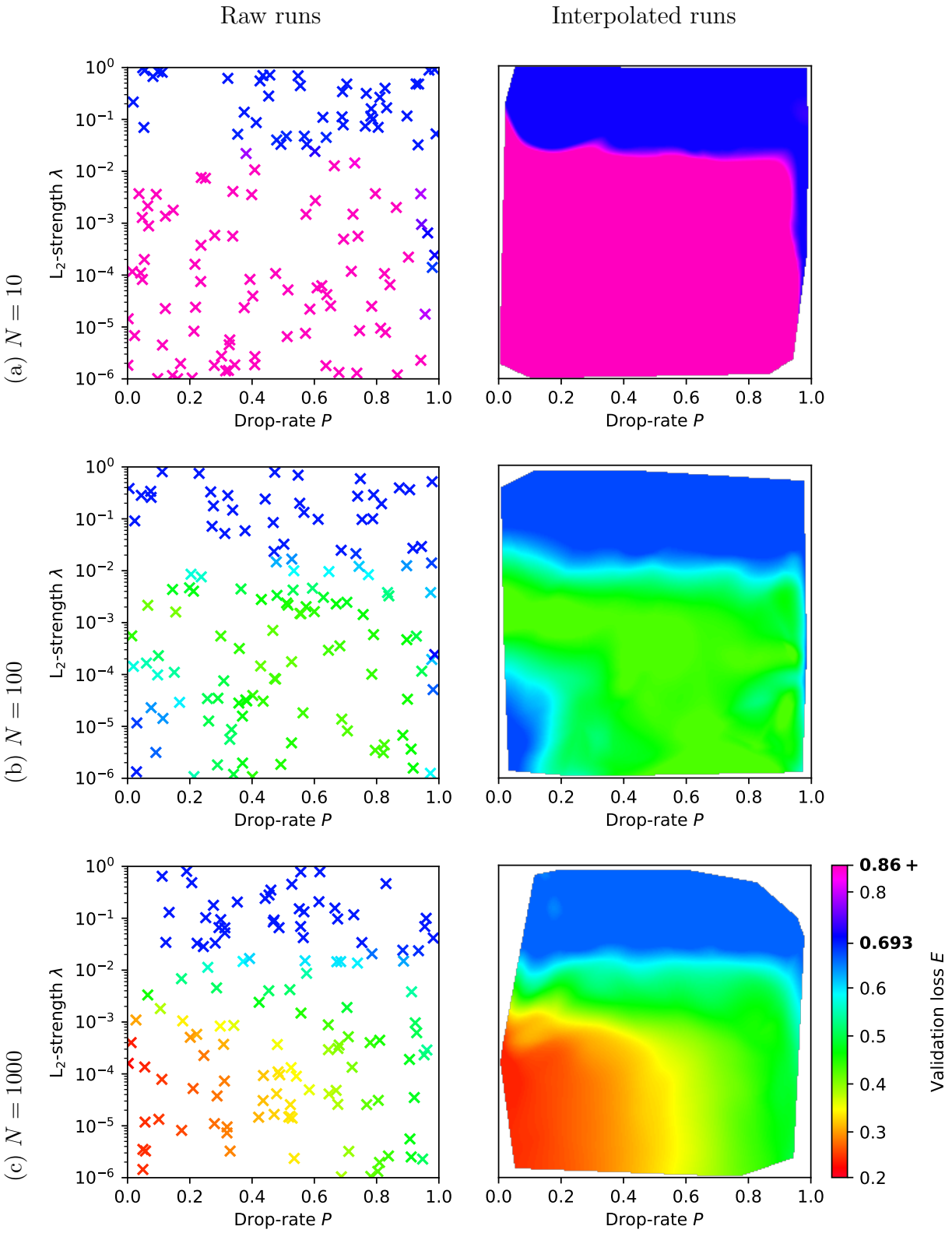
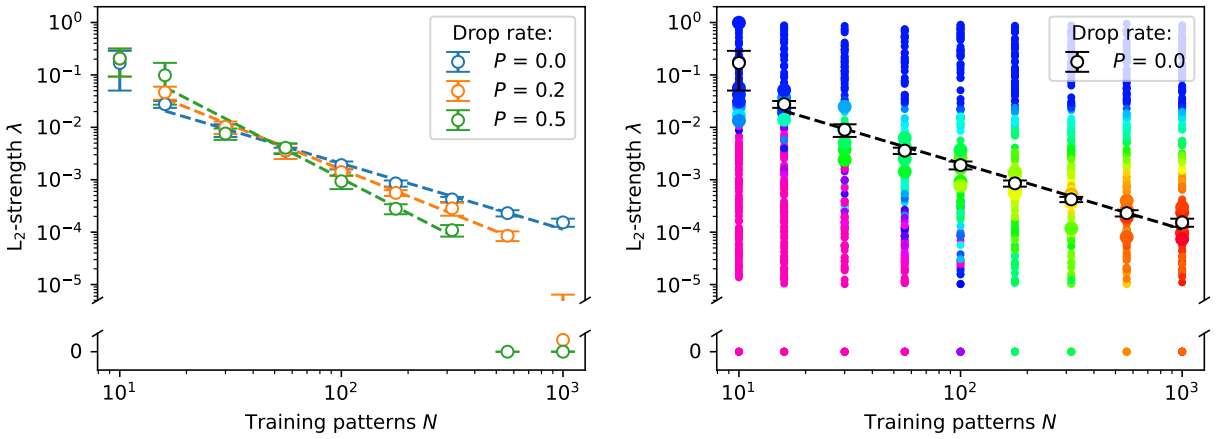
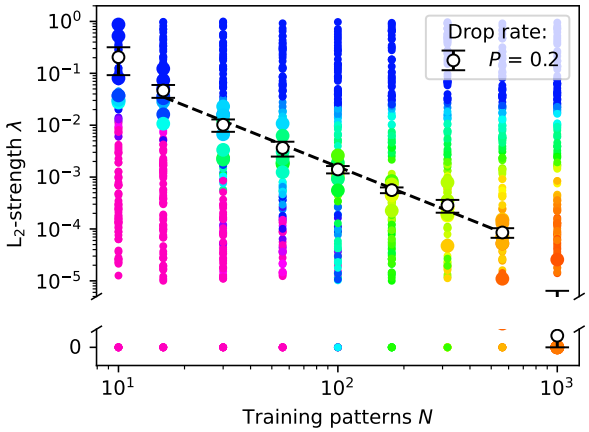


Figure 4.1: Random hyperparameter search heat maps of L_2 strength and drop rate for various amounts of training patterns N, coloured by validation loss for 4000 epochs

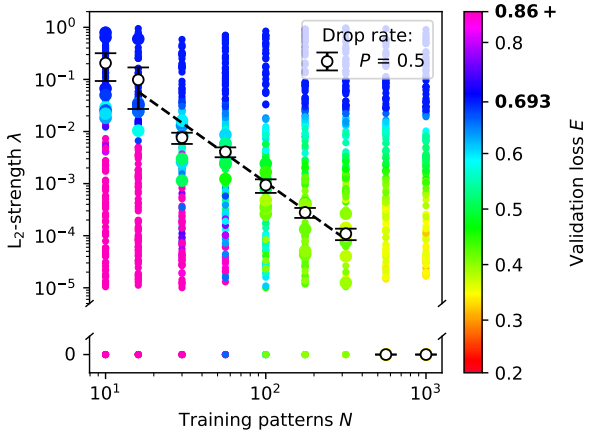


(a) Various drop rates P

(b) $P = 0.0$



(c) $P = 0.2$



(d) $P = 0.5$

Figure 4.2: Plots of mean optimal L_2 strength against number of training patterns, with standard errors, coloured by validation loss for 4000 epochs

Drop rate, P	**Gradient**
0.0	-1.26 ± 0.05
0.2	-1.69 ± 0.06
0.5	-2.19 ± 0.18

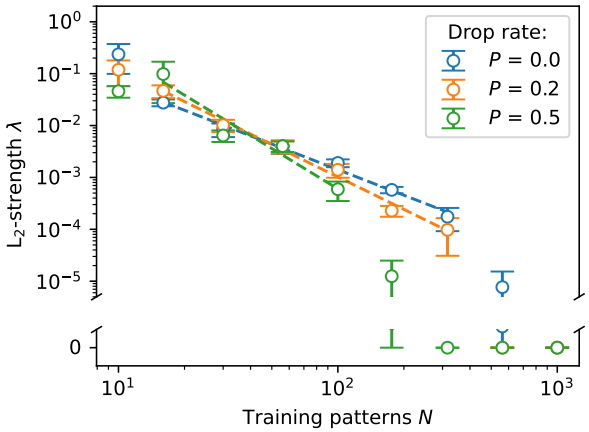
Table 4.1: Linear regression gradients, with standard errors, of mean optimal L_2 strength against number of training patterns for 4000 epochs

Within suitable regions of N, least-squares linear regressions were fit to $\log\langle\lambda^*\rangle$. Presented in table 4.1 are the properties of these regressions with their standard errors, for which it was suggested that the gradient steepened with increasing P. Data suggested that, above some critical number of training patterns, it was optimal to have no L_2 reg-

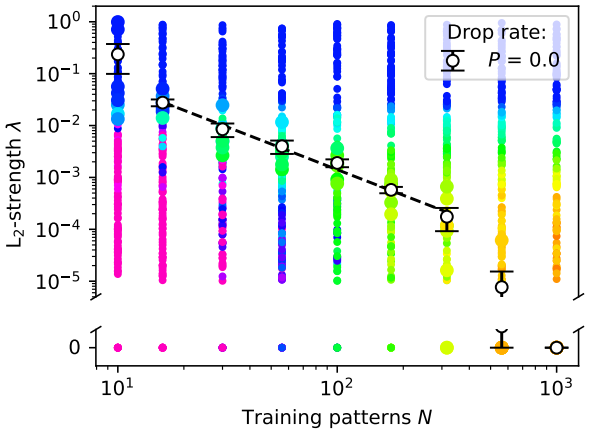
ularisation, as demonstrated in figures 4.2c and 4.2d. Data suggested that this critical number of training patterns was inversely proportional to the drop rate P.

4.2 Reduced epochs for early stopping

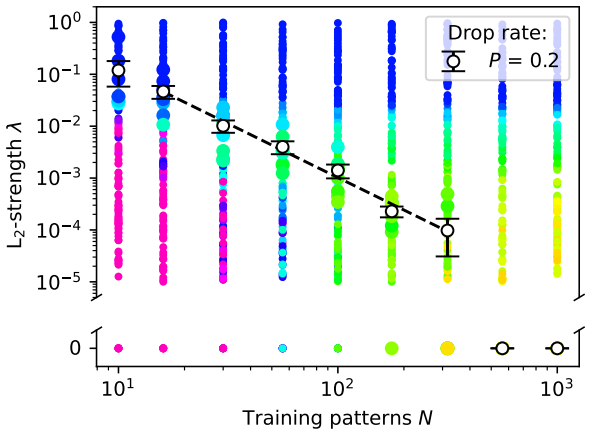
By reducing the number of epochs from 4000 to 1000, early stopping was introduced for runs with approximately $N > 10^{2.5}$. Conversely, for runs with approximately $N \leq 10^{2.5}$, whilst overfitting was not completely eliminated, the amount of overfitting, and hence need for regularisation, was reduced. This allowed for the exclusive observation of both the effects of early stopping, and the effects of reduced overfitting, on optimal λ's relation to N.



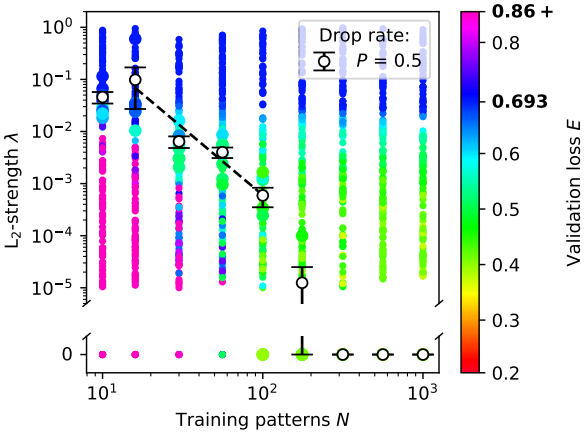
(a) Various drop rates P



(b) $P = 0.0$



(c) $P = 0.2$



(d) $P = 0.5$

Figure 4.3: Plots of mean optimal L_2 strength against number of training patterns, with standard errors, coloured by validation loss for 1000 epochs

Drop rate, P	**Gradient**
0.0	-1.64 ± 0.08
0.2	-2.08 ± 0.11
0.5	-2.58 ± 0.47

Table 4.2: Linear regression gradients, with standard errors, of mean optimal L_2 strength against number of training patterns for 1000 epochs

Concurrence of minimum validation loss with the critical N above which optimal λ was 0, as demonstrated in figures 3.2 and 4.3 respectively, suggested that the occurrence of early stopping is what determined the upper limit of the hypothesis. This was likely described by early stopping removing any need for regularisation, as overfitting was already eliminated. Hence, with early stopping, it was optimal to have no L_2 at all. For this reduced number of epochs, over the working region of the hypothesis, table 4.2 displayed steeper gradients in comparison to table 4.1.

5

Discussion

In this thesis, I introduced and tested a predictor for optimal L_2 strength, and considered how the further involvement of dropout and early stopping may affect the relation. When ANNs were faced with substantial overfitting, the results supported the hypothesis that L_2 strength's optimal value is inversely proportional to the number of training patterns within a region of typical usage. The success of the hypothesis was best visualised in figure 4.2.

With the introduction of dropout, the constant of proportionality was found to increase, as there was an intuitive reduction in optimal L_2 strength for large data sets containing many training patterns. This finding was further supported by figure 4.1, which demonstrated how regions of optimal regularisation rely on a combination of L_2 and dropout, with an increase in one regularisation technique requiring a complementary decrease in the other. Low variation of validation loss in the immediate vicinity of optimal regularisation suggested that, once a region of optimal regularisation begins to appear, excessive fine-tuning of regularisation strengths is not worthwhile, as demonstrated by figure 4.1.

From cross referencing figures 3.2, 4.2 and 4.3, it was consistently observed that, when the presence of early stopping was expected, optimal L_2 strength was 0. This was likely

a consequence of early stopping causing an absence of overfitting, such that there was a lack of a need for regularisation. When L_2 and early stopping were further combined with dropout, data demonstrated the working region of the hypothesis was further reduced.

Due to this study being limited to a single binary classification problem, the quantitative aspect of the findings are quite impractical on their own. Indicated in figure 4.1 and in contrast to the findings in [4], minimum validation loss was suggested to have an insignificant dependence on the choice of regularisation techniques. That is, there was no clear benefit to generalisation performance when using a combination of both dropout and L_2, compared to the independent use of either technique. This may suggest that our synthetic data was a poor representation of real world data, or perhaps that an architecture known to favour dropout, such as a deep network, may have been more favourable.

With these limitations of the study in mind, the success of the hypothesis should not be overlooked. If the results were found to generalise to other data sets, then the predictor for optimal L_2 strength may find real-world usage for estimating a ball-park range over which the optimal L_2 strength lies, prompting the application of random hyperparameter search over a reduced range. In particular, the proportionality of optimal L_2 strength to the number of training patterns, for a given problem and architecture, could be calculated over a range of few training patterns, allowing the optimal L_2 strength to be extrapolated for a greater number of training patterns in a computationally inexpensive manner.

Acknowledgements

Many thanks to Patrik Edén for his helpful and kind supervision, Mattias Ohlsson for his supplementary support, and to my mum for her proof-reading. Much appreciation to Alexander Degener, Oskar Bolinder & Rasmus Sjöö, with whom I worked closely.

Bibliography

[1] Stuart Russell. Banning Lethal Autonomous Weapons: An Education. *Issues in Science and Technology*, 38(3):60–65, 2022.

[2] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-Art in Artificial Neural Network Applications: A Survey. *Heliyon*, 4(11):e00938, 2018.

[3] Mattias Ohlsson and Patrik Edén. *Introduction to Artificial Neural Networks and Deep Learning*. Computational Biology and Biological Physics, Department of Astronomy and Theoretical Physics, Lund University, 2021.

[4] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[5] Ekachai Phaisangittisagul. An Analysis of the Regularization Between L2 and Dropout in Single Hidden Layer Neural Network. *7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS)*, pages 174–179, 2016.

[6] James Bergstra and Yoshua Bengio. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13(10):281–305, 2012.

[7] Anders Krogh and John Hertz. A Simple Weight Decay Can Improve Generalization. In J. Moody, S. Hanson, and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4. Morgan-Kaufmann, 1991.

[8] Mario Figueiredo. Adaptive Sparseness for Supervised Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1150–1159, 2003.

[9] Hava Siegelmann and Eduardo Sontag. Turing computability with neural nets. *Applied Mathematics Letters*, 4:77–80, 06 1997.

[10] Brady Neal, Sarthak Mittal, Aristide Baratin, Vinayak Tantia, Matthew Scicluna, Simon Lacoste-Julien, and Ioannis Mitliagkas. A Modern Take on the Bias-Variance Tradeoff in Neural Networks. *CoRR*, abs/1810.08591, 2018.

[11] Brian Keng. A Probabilistic Interpretation of Regularization. *Bounded Rationality*, 2016.

[12] Pierre Baldi and Peter Sadowski. Understanding dropout. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[13] Andrew Ng. Orthogonalization. *DeepLearning.AI: Course 3, Week 1, Lecture 2: Introduction to ML strategy*, 2017.

[14] Joseph Binns, Alexander Degener, Oskar Bolinder and Rasmus Sjöö. Artificial Neural Networks Python3 code base. GitHub, 2022.

[15] John Hunter. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.