LU-TP 22-29
June 2022

Prediction of appropriate L2 regularization strengths through Bayesian formalism

Alexander Degener

Department of Astronomy and Theoretical Physics, Lund University

Bachelor thesis supervised by Patrik Edén

Abstract

This paper proposes and investigates a Bayesian relation between optimal L2 regularization strengths and the number of training patterns and hidden nodes used for an artificial neural network. The results support the proposed dependence for number of training patterns, while the dependence on hidden architecture was less clear. Finally, applying different regularization strengths on different layers, rather than the same on all, resulted in better validation performances. The essential programs for training ANNs were developed for these studies, along with functionality for synthetic data generation, which together provided a controlled and flexible environment.

Popular Science Description

Imagine you are trying to come up with a plan for solving a problem, but the best strategy you can come up with is to test every possible approach. This could work if you have as many tries as you want, but it would take quite some time and if you eventually settle for an approach you will likely not know if there exists a better one. Analyzing the problem to instead figure out an appropriate approach given its nature can be both more efficient and effective. In this study, we analyze some strategies for machine learning, hoping to reduce the need for a trial-and-error approach.

Machine learning can perform well on certain tasks that would be very demanding to solve in alternative ways. An example of such a task is finding relations between medical data of individuals and their health-conditions. This can be done through machine learning of the type "Artificial Neural Networks" also known as ANNs, that use a specific kind of models inspired by the neurons in a human brain. These models are implemented as programs that can be trained through many iterations of adaptation.

When training the program to solve difficult problems, we often use some information about them to set parameter values for a good starting point. One parameter currently laborious to determine is called the L2 regularization strength, which helps make the program more able to handle other similar tasks. Our research aims to find a way to set this parameter, given certain knowledge about the initial problem.

A common practice for finding a useful L2 regularization strength is through trial and error, which can yield quite useful results, given enough time. We would like to reduce the time required by finding at least a narrower range in which the most appropriate value lies. To do this, a proposed dependence for L2 regularization strength on data size and ANN size is investigated.

Abbreviations

ANN:	Artificial Neural Network
CEE:	Cross entropy error
MLP:	Multilayer Perceptron
MSE:	Mean squared error
GD:	Gradient Descent
SGD:	Stochastic Gradient Descent

Acknowledgements

I would like to thank my supervisor, Patrik Edén, for his positive and effective guidance throughout the whole project. Working together was a great learning experience. I would also like to thank Joseph Binns, Oskar Bolinder and Rasmus Sjöö for developing the base programs together with me. It was a fun part of the project and I enjoyed our various discussions. Finally, I thank Mattias Ohlsson for his additional input, which was very appreciated.

Contents

1 Introduction

ANN models learn by training on data. During this training, parameters called weights are adjusted for the model to perform better on the data set. Each weight influences how much certain data or data-combinations are used to find the solution. However, a model can be overtrained, which means that it has been trained to perform well on a single data set, but not on other data from the same problem.

A flexible model, one that can adjust the weights freely to fit a specific data set, is more prone to become overtrained. Therefore, an effective generalization method is to introduce a regularization term. This causes weights of great magnitude to significantly increase the loss, the measure which the model is trained to minimize during training.

One of the most common and effective regularization terms is called the L2 norm, and just like all other regularization terms it comes with a regularization strength, determining how much it impacts the training. The regularization strength is set as a hyperparameter during model selection [1].

Before training, ANN models are adapted to the problem through hyperparameter settings based on its characteristics. The process of creating the model and finding appropriate hyperparameters is arguably the most time-consuming part of working with ANNs, as strong models often use several hyperparameters that cannot be adapted independently and have to be set by trial and error [2]. Therefore, being able to predict an appropriate L2 regularization strength would be useful in reducing the time required to find an acceptable model. It might also highlight properties of the problem based on the initialization of the model.

The initialization of the weights play an important role for training [3], but possibly also for how the regularization strength should be chosen. Using a Bayesian formalism, it could be possible to relate the initial assumptions on the weights to the best L2 regularization strength. This paper will present and discuss research conducted to test a hypothesis for finding an appropriate L2 regularization strength without trial and error.

2 Theory and hypothesis

2.1 The ANN

Inspired by the neurons in a human brain, the ANN is a model which can take inputs, process them, and produce an output, or interpretation, which is meaningful in some context. Since the model is deterministic and less complicated than actual neurons it is more limited, but also easier to explain. Figure 1 shows an ANN with four input nodes, one **hidden layer** of five **hidden nodes**, and two output nodes.

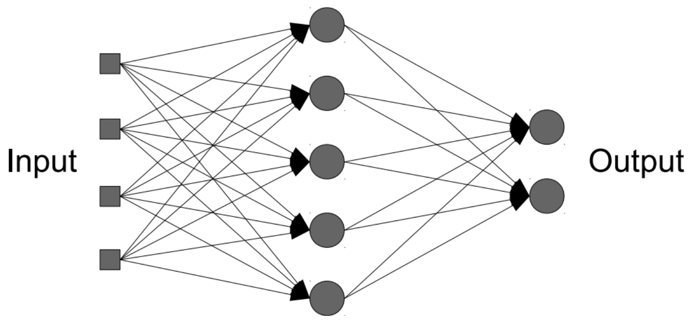


Figure 1: Fully connected Multilayer Perceptron, MLP, with one hidden layer, reproduced from [1] with permission.

The ANN is supposed to solve a problem represented by **patterns** that all consist of an input which will be fed into the ANN, and a target which the ANN will be trained towards outputting. The weights, $\boldsymbol{\omega}$, are trained over multiple **epochs**, each updating the weights slightly through back propagation [4]. A common performance measure for binary classification problems is cross entropy error, CEE, where the network is trained to minimize the loss function.

$$E(\boldsymbol{\omega}) = -\frac{1}{N}\sum_n \left[d_n \ln{(y_n)} + (1-d_n)\ln{(1-y_n)}\right] \tag{2.1}$$

In this equation, N is the number of patterns. Each pattern n has target d_n and ANN output element $y_n = y(\mathbf{x}_n, \boldsymbol{\omega})$, where $\mathbf{x}_n$ are the inputs for the pattern. For binary classification, there is only one output node.

2.2 Maximum likelihood

Minimizing the cross entropy error corresponds to maximizing the likelihood for the output to match the target values from the data set $\mathcal{D}$, i.e. maximizing $p(\mathcal{D} \mid \boldsymbol{\omega})$. An expression

$p(a|b)$ signifies conditional probability for a given b. Assuming independent samples the distribution can be expressed as

$$p(\mathcal{D} \mid \boldsymbol{\omega}) = \prod_n p\left(d_n, \mathbf{x}_n \mid \boldsymbol{\omega}\right) = \prod_n p\left(d_n \mid \mathbf{x}_n, \boldsymbol{\omega}\right) p\left(\mathbf{x}_n\right) \tag{2.2}$$

It is preferable to optimize a sum rather than a product, and for this purpose the negative logarithm of equation (2.2) can be minimized:

$$-\ln p(\mathcal{D} \mid \boldsymbol{\omega}) = -\sum_{n=1}^{N} \ln p\left(d_n \mid \mathbf{x}_n, \boldsymbol{\omega}\right) - \sum_{n=1}^{N} \ln p\left(\mathbf{x}_n\right) \tag{2.3}$$

As the last sum is independent of $\boldsymbol{\omega}$, the loss function to minimize can be expressed like

$$E(\boldsymbol{\omega}) \propto -\sum_n \ln p\left(d_n \mid \mathbf{x}_n, \boldsymbol{\omega}\right) \tag{2.4}$$

For binary classification, $\ln p\left(d_n \mid \mathbf{x}_n, \boldsymbol{\omega}\right) = d_n \ln(y_n) + (1 - d_n)\ln(1 - y_n)$. Introducing a proportionality constant $1/N$, equation (2.4) is identical to the CEE in equation (2.1).

2.3 Bayesian formalism

The above ideas can be extended to a Bayesian framework. When a model is built there are made assumptions about the weights $\boldsymbol{\omega}$, such as number of weights and their initial values. The way the number of weights and their initialization is decided can be seen as a **prior distribution** $p(\boldsymbol{\omega})$. Bayes' theorem [5] gives that we have

$$p(\boldsymbol{\omega} \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \boldsymbol{\omega}) p(\boldsymbol{\omega})}{p(\mathcal{D})} \tag{2.5}$$

$$p(\mathcal{D}) = \int p(\mathcal{D} \mid \boldsymbol{\omega}) p(\boldsymbol{\omega}) d\boldsymbol{\omega} \tag{2.6}$$

The posterior $p(\boldsymbol{\omega} \mid \mathcal{D})$ tells us about the assumptions on $\boldsymbol{\omega}$ given the data set $\mathcal{D}$. Like the case for equation (2.3) it is preferred to optimize a sum, and we can minimize the following expression to find $\boldsymbol{\omega}$ with the largest posterior:

$$-\ln p(\boldsymbol{\omega} \mid \mathcal{D}) = -\ln p(\mathcal{D} \mid \boldsymbol{\omega}) - \ln p(\boldsymbol{\omega}) + \ln p(\mathcal{D}) \tag{2.7}$$

Removing all terms independent of $\boldsymbol{\omega}$, as they can be disregarded in minimization, and assuming independent samples like for equation (2.2), the function to minimize can be expressed as

$$S(\boldsymbol{\omega}) = -\sum_n \ln p\left(\mathbf{d}_n \mid \mathbf{x}_n, \boldsymbol{\omega}\right) - \ln p(\boldsymbol{\omega}) \tag{2.8}$$

Thus, a prior distribution for the weights leads to an extra loss term which depends on $\boldsymbol{\omega}$ but not on the data set.

2.4 The L2 norm

Equation (2.8) can be expressed as

$$\tilde{E}(\boldsymbol{\omega}) = E(\boldsymbol{\omega}) + \lambda\Omega(\boldsymbol{\omega}) \tag{2.9}$$

where $\Omega(\boldsymbol{\omega})$ is called a regularization term, and λ the regularization strength. A common regularization term is the **L2 norm**:

$$\Omega(\boldsymbol{\omega}) = \frac{1}{2}\sum_j \omega_j^2 \tag{2.10}$$

This term counteracts large weights and has a greater impact the bigger λ is.

2.4.1 Hypothesis

Combining equations (2.1), (2.8) and (2.9) gives

$$\lambda\Omega(\boldsymbol{\omega}) = -\frac{1}{N}\ln p(\boldsymbol{\omega}) \tag{2.11}$$

This suggests that a successful λ should be proportional to $1/N$.

Furthermore, we hypothesize that a good **initialization scheme** represents a good prior $p(\boldsymbol{\omega})$. All weights must be given initial values at the beginning of training so that the first output can be produced. Here, a good initialization is important for successful training. One of the most common initialization methods is to choose a weight ω_{ij} from node j to node i from a Gaussian distribution with zero mean and variance $1/K_i$, where K_i is the number of inputs to node i [1][3]. For a **fully connected** MLP, K_i is equal to the number of nodes in the previous layer. Under our hypothesis, this initialization corresponds to a **Gaussian prior**:

$$p(\boldsymbol{\omega}_i) \propto \exp\left(-\frac{1}{2\sigma_i^2}\sum_j \omega_{ij}^2\right) \tag{2.12}$$

Using this as the prior, and assuming $\sigma_i^2 \propto 1/K_i$, the $1/N$-dependence implied by equation (2.11) is modified to node-specific optimal L2 strengths

$$\lambda_i = \frac{1}{\sigma_i^2 N} \propto \frac{K_i}{N} \tag{2.13}$$

This relation is the hypothesis to be studied. It suggests that optimal L2 regularization could be achieved if the strength is allowed to vary between different nodes depending on K_i and N.

3 Method

As equations 2.11 and 2.13 regards successful λ values, the definition of successful becomes relevant. Let λ^* be defined as the λ which yields the best validation performance for a certain model and number of training patterns. To test the hypothesis, it will be investigated whether there exists a correlation between λ^* and different numbers of training patterns or hidden nodes, and what such a correlation would look like.

3.1 Model specifications

The goal of this study is to find λ^* for models where the following parameters were varied:

- *Number of training patterns*, varied in the range 10 to 1000.
- *Number of hidden nodes in one hidden layer*, varied in the range 1 to 200.
- *Number of hidden layers*, varied in the range 1 to 6.
- *Values of λ for different layers*, varied over the values in Table 3.

All results were produced using the following specifications.

Initialization
The weights were randomly initialized from a Gaussian with variance $1/K_i$ and zero mean. Here, K_i is the number of non-bias inputs from the previous layer, equaling number of dimensions for the input layer and number of hidden nodes for all other layers.

Activation functions
The following activation functions were used:

$$\text{Logistic:} \qquad \varphi_0(a) = \frac{1}{1+e^{-a}} \tag{3.14}$$

$$\text{Hyperbolic Tangent:} \qquad \varphi_H(a) = \tanh(a) \tag{3.15}$$

Logistic was used as the output activation function and Hyperbolic Tangent for all other layers.

Weight updating
The weights were updated using regular gradient descent and batch updating, with a learning rate of 0.1. Stochastic Gradient Descent, SGD, was ultimately not used since it can regularize, making it harder to see the connection between the L2 regularization strength and other parameters. To keep the number of inputs K_i constant, and avoid additional regularization through dropout [6], the networks were also kept fully connected.

Performance measure
CEE, equation 2.1, was used to determine λ^*. It was also studied how accuracy, the fraction of correct predictions, was affected by varying λ-values.

3.2 Data sets

To be able to train the model and evaluate its performance, data is required. Data sets were synthetically generated to be at a difficulty where an MLP with one hidden layer of about 20 hidden nodes could solve it acceptably well, given about 100-200 patterns, only using regular gradient descent, L2 regularization and preferably fewer than 5000 epochs. Here, a validation performance for loss of about 0.4 was accepted.

Binary classification was chosen as the task, since the generated patterns then were simple to judge in terms of difficulty. A program was developed that could produce two Gaussian clouds, each making up one class of data points. The differences in performance by a fixed model were then compared when varying the number of dimensions of the clouds, their sizes, and the size of overlap between the clouds. After this, a data set distribution called "D1" was chosen for the first study; number of training patterns. The properties of this distribution can be seen in Table 1, and Figure 2 shows an example of a data set synthetically generated using those settings, plotted in two dimensions.

name	D1
input dimensions	10
balanced classes	2
distance between cloud centers	4
cloud width per dimension	4 and 2

Table 1: Properties of the first chosen data set distribution

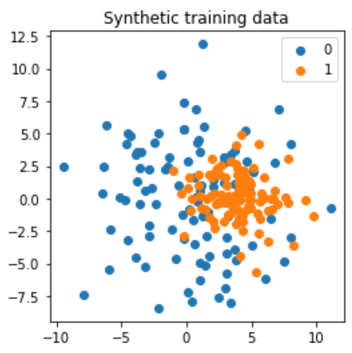


Figure 2: Data set of 200 patterns from D1. Two out of ten input dimensions are shown.

Synthetically generating the data meant that as many validation patterns as desired could be created, which allowed the λ^* validation performance to be of high certainty. In the same way, different sets of training data could be produced from the chosen distribution, making it possible to obtain a mean and standard error of λ^*.

Additionally, it allowed for the creation of a data set which made the ANN performance more dependent on number of hidden nodes. The idea was that this would make it easier to study how a change in number of hidden nodes or hidden layers related to λ^*. For this reason, the overlap and relative densities of the clouds were changed for a second data set, "D2", such that more hyperplanes would be required to distinguish the classes. D2 was created and tested with the same methods as D1.

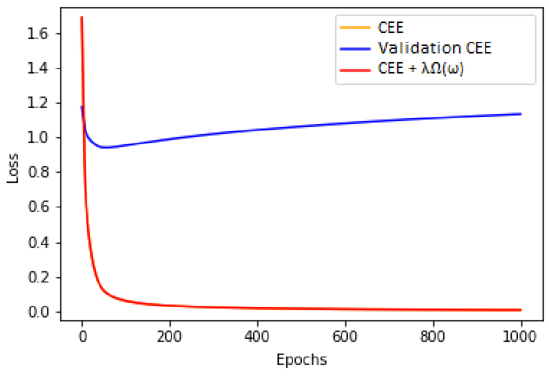
name	D2
input dimensions	10
balanced classes	2
distance between cloud centers	4
cloud width per dimension	32 and 8

Table 2: Properties of the second chosen data set distribution

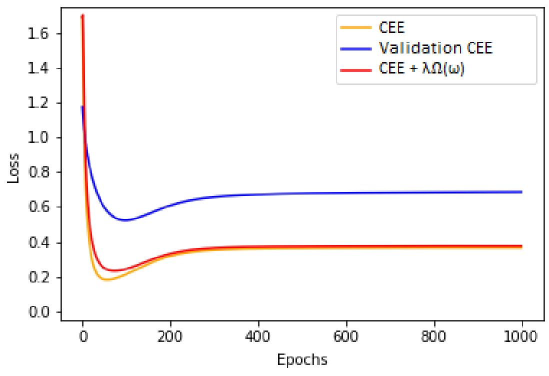
For the studies on pattern dependence of λ^* in sections 4.2 and 4.4, the data set distribution D1 in Table 1 was used. All other results were produced using the distribution D2 in Table 2. The number of generated validation patterns was 2000, independent of which parameters were studied.

3.2.1 Achieving overtraining

The data sets in this study should lead to overtraining in the absence of regularization, so that there is a notable effect of λ to be found. This was confirmed by plotting the loss over epochs as can be seen in Figure 3.



(a) $\lambda = 0$



(b) $\lambda = 0.01$

Figure 3: The red plot shows loss with L2 regularization, which the model was trained to minimize, and the yellow plot shows only the CEE contribution to this loss. The blue plot shows the models validation performance. Having $\lambda = 0$ in (a) makes the red and yellow plots overlap. Training was done with 10 patterns from distribution D1 and 20 hidden nodes in one layer.

3.3 Finding the best L2 regularization strength, λ^*

When varying one of the parameters to investigate, λ^* was obtained in the following way:

1. Try different values for the parameter P to investigate.
2. For each value of P try a range of values for λ, presented in Table 3.
3. The λ that resulted in the highest validation performance is then saved as λ^*, for that value of P.

This procedure for finding λ^* was conducted nine times for each combination of settings. Each time was for a different generated data set from the the chosen distribution, allowing a mean and sample variance of λ^* to be obtained. The variance was calculated using mean squared error, MSE, and the mean along with standard error of λ^* were plotted against the varied parameter. The range for λ was adapted such that the resolution is high enough to assess the hypothesis, and can be found in Table 3.

λ values
[1e-5, 2e-5, 4e-5, 0.0001, 0.0002, 0.0004, 0.001, 0.002, 0.004, 0.01, 0.02, 0.04, 0.1, 0.2, 0.4]

Table 3: Tested values for λ.

3.4 Training properly

Changing hyperparameters such as number of hidden nodes or hidden layers can have a great effect on the number of epochs it takes for the model to finish training. Due to limitations in available computational power and time, the number of epochs were not set exceedingly large, but instead to a number found appropriate through trial and error. It is important to have a sufficient number of epochs when determining λ^*, as the best λ found otherwise could be affected by **early stopping**, which is another type of regularization.

To confirm that the program finished training, plots of the same type as those in Figure 3 were produced for the model being investigated. Additionally, the mean and standard error of the final validation performance for nine different data sets of the chosen distribution was produced and plotted.

3.5 Program development

To ensure control and flexibility for the investigations, a program was developed instead of using existing libraries for ANNs such as Keras [7]. The development was conducted in a group of four people. Tasks were divided to create different functionalities in a manner possible by the object oriented nature of Python. Merging of code was done on a dedicated Github branch. This collaboration continued until functionality for the following was working:

- Synthetic data generation for classification problems
- Training through backpropagation with L2 regularization
- Conducting SGD
- Calculating performance measures such as CEE and accuracy
- Identifying and plotting λ^* over number of patterns
- Calculating and plotting mean with error bars for different data sets

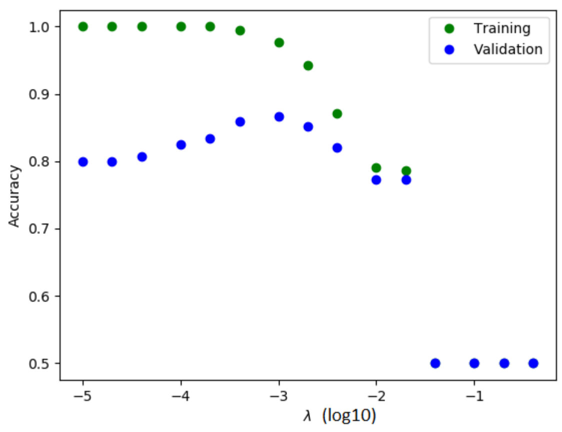
After the collaboration, the code was updated to fit the needs for additional studies such as finding λ^* when varying number of hidden nodes, number hidden layers and having different λ values on different layers.

4 Results

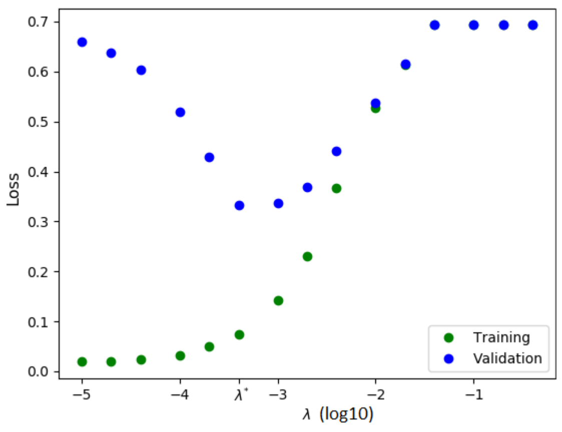
4.1 Verification

The plots for λ^* over varying parameters were always produced using nine data sets from the distribution chosen for that study. Proper training for all results were confirmed through three types of figures:

- A plot of training loss, training loss including L2, and validation loss over epochs, as exemplified in Figure 3 (b)
- A plot of training and validation accuracy over the tested lambdas values, as exemplified in Figure 4 (a)
- A plot of final training and validation loss over the tested lambdas values, as exemplified in Figure 4 (b)



(a) Accuracy over λ after finished training



(b) Loss over λ after finished training

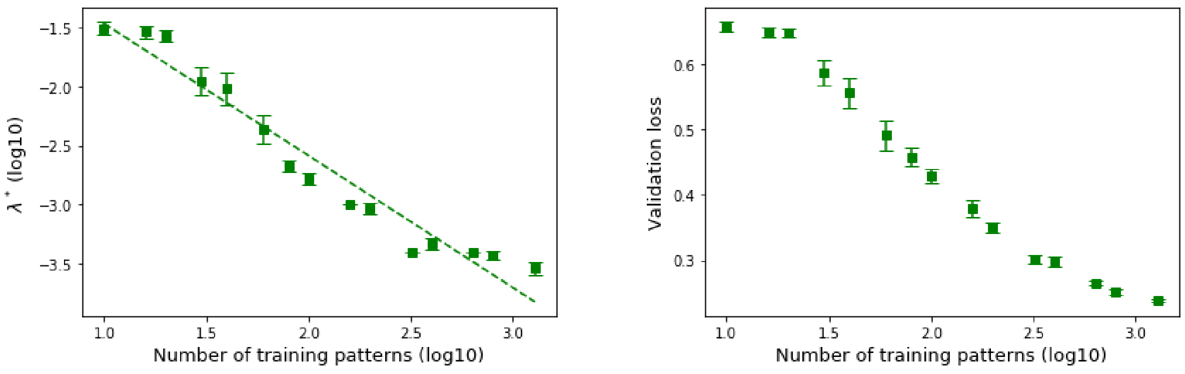
Figure 4: Accuracy and loss for 210 training patterns and 20 hidden nodes in one layer for different λ values. In this example, $\lambda^* = 0.0004$ when optimized for validation loss. Note that (a) and (b) have log10-scales for λ values.

The figures 4 (a) and (b) both show that the model has a varying performance dependent on λ, and that λ^* can be distinguished. The difference in validation performance is relatively small for some values of λ close to λ^*, suggesting that slightly unfinished training could affect the choice of λ^*.

4.2 Number of training patterns

The first parameter studied was the number of training patterns as it was deemed the most likely to produce results agreeing with the hypothesis and thus would indicate whether it was promising to study the other parameters as well.

Two plots produced along with the verification results in Figure 4 can be found in Figure 5, displaying λ^* over number of training patterns, and validation performance for each data point.



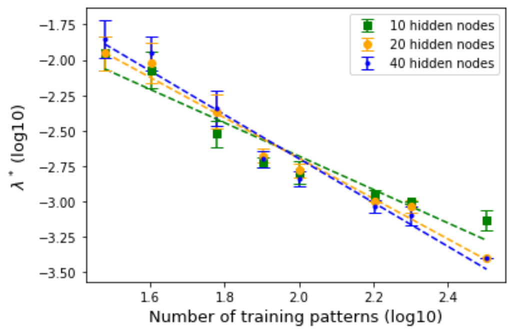
(a) λ^* over number of training patterns (b) Validation performance

Figure 5: λ^* and validation performance over number of training patterns. Produced using 5000 epochs and 20 hidden nodes in one layer.

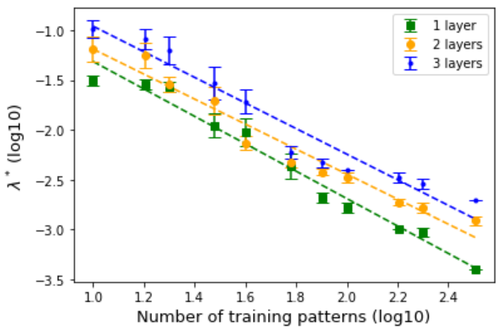
A slope of -1.1 is obtained when performing linear regression on the data points in the log-log-plot from Figure 5 (a). This translates rather well to a $1/N$ behaviour of λ^*. Effects on this dependence when modifying network architecture can be seen in Figure 6.

The effects of having different amounts of hidden nodes were unclear, while the slopes varied as can be seen for the plots in Figure 6 (a). These slopes are steeper than the one in Figure 5, being at least $-1.2 < -1$. Even the slope for the same number of hidden nodes was steeper, which indicates that the trend is sensitive to which interval of pattern numbers is studied.

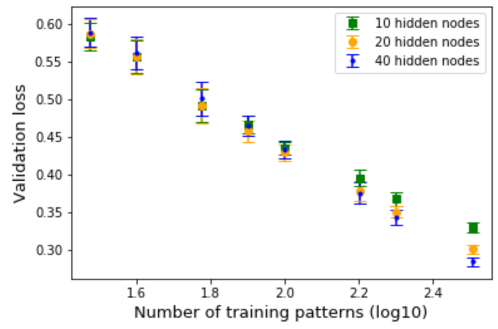
In Figure 6 (b) the plots show a trend of higher λ^* for higher number of hidden layers, which could be related to the large increase in number of trainable parameters. However, the hypothesis does not predict such an increase in λ^*, and the effects of changing number of hidden layers were investigated further.



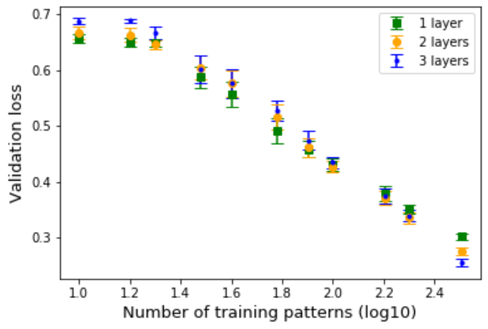
(a) λ^* over number of training patterns for different numbers of hidden nodes. The slopes ranged from -1.2 to -1.5.



(b) λ^* over number of training patterns for different numbers of hidden layers. The slopes ranged from -1.25 to -1.4.



(c) Validation performance for (a)



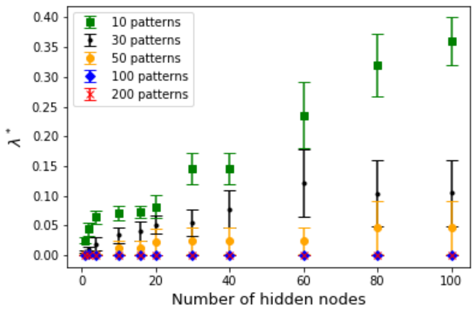
(d) Validation performance for (b)

Figure 6: λ^* over number of training patterns and validation performance. The plots in (a) were produced using 5000 epochs and different numbers of hidden nodes in one layer. The plots in (b) were produced using 10000 epochs, 20 hidden nodes per layer, and different numbers of hidden layers.

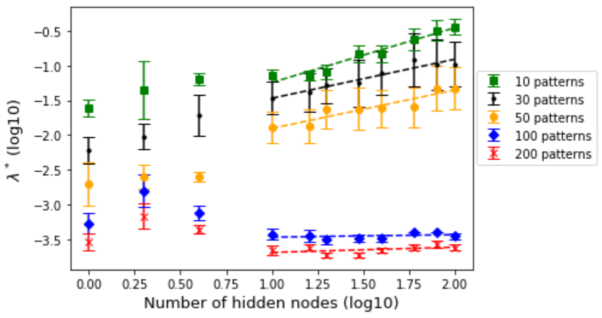
4.3 Number of hidden nodes and hidden layers

Number of hidden nodes

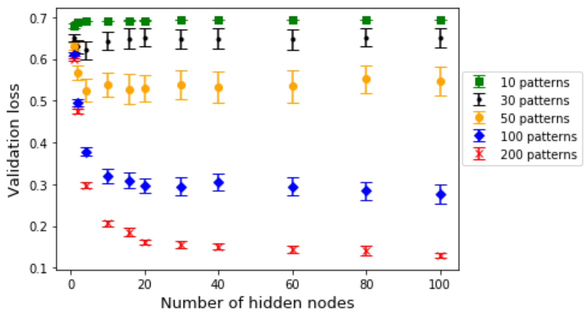
Obtaining clear results when varying number of hidden nodes proved to be more difficult, as changing that parameter affects training of the model more drastically than changing number of training patterns within the ranges deemed appropriate for this study. The ranges were made such that the model could solve the problem with validation loss below 0.6, for most of the data points.



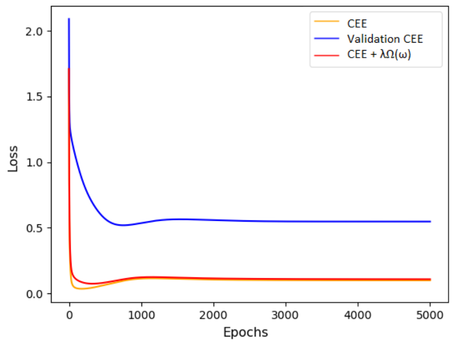
(a) λ^* over number of hidden nodes, plotted linearly. The points for 100 patterns and 200 patterns appear to overlap for this scale



(b) λ^* over number of hidden nodes, plotted log-log. The regression for 10 patterns has slope +0.79, 50 patterns has slope +0.55 and 100 patterns has a slope of +0.035.



(c) Validation performance for (a) and (b)



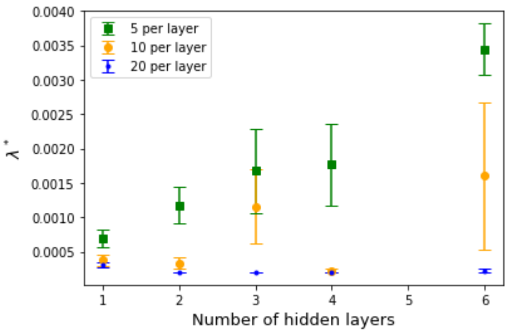
(d) Training and validation performance over epochs for the data point: 50 patterns, 100 hidden nodes

Figure 7: Results for λ^* over number of hidden nodes. Produced using 5000 epochs and different numbers of training patterns. Linear regressions have been made in (b) for the number of hidden nodes that produced consistent validation performance.

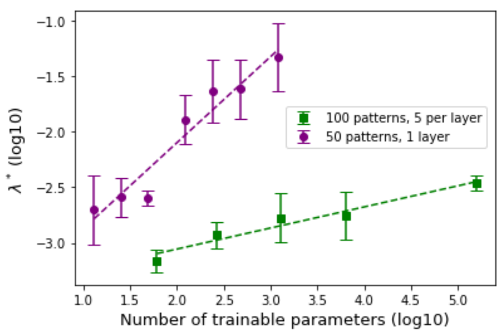
To ensure proper training, plots of loss over epochs as in Figure 3 were produced for more data sets per result in this study. Figure 7 shows plots for varying number of hidden nodes, with five different numbers of training patterns. Of the plots in Figure 7 (a), the one for 50 patterns has the largest data set for which L2 regularization is needed. The hypothesis suggests a linear relation $\lambda \propto K_i$, which corresponds to a slope of +1 in the log-log plot. Such a slope was not expected for these results, as they were produced using a common λ-value for all layers.

Number of hidden layers

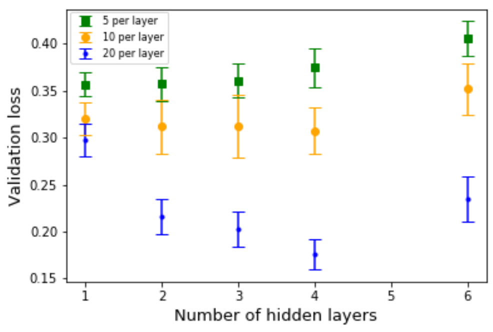
Relating λ^* directly to the number of hidden layers is not something the hypothesis suggests is possible. However, Figure 8 (a) shows that λ^* is not independent of that parameter.



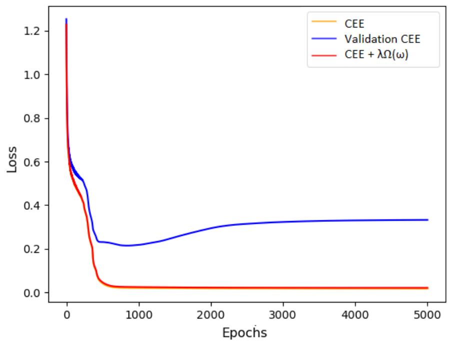
(a) λ^* over number of hidden layers for different numbers of hidden nodes per layer. Produced using 5000 epochs and 100 training patterns



(b) λ^* over number of trainable parameters. The regression for 100 patterns, 5 per layer has slope +0.19 and for 50 patterns, 1 layer has slope +0.78.



(c) Validation performance for (a)

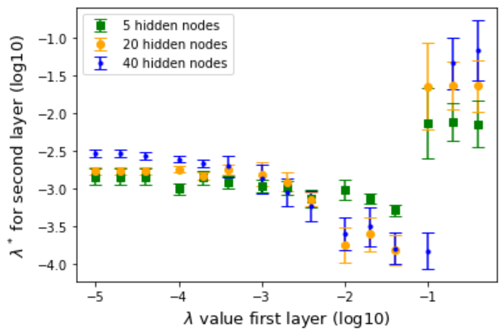


(d) Training and validation performance over epochs for the data point: 5 hidden nodes per layer, 6 hidden layers

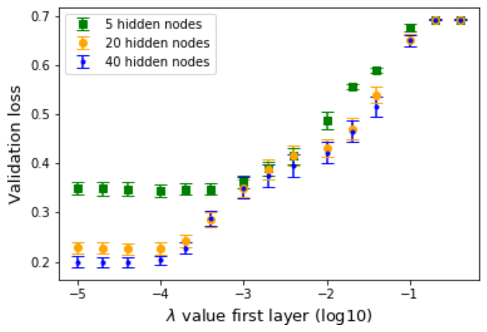
Figure 8: Results for λ^* over number of hidden layers and in (b) a comparison between a plot from (a), 5 per layer, and from Figure 7 (a), 50 patterns, 1 layer.

A comparison can be made using number of trainable parameters, which is number of weights in the models. The results for such a comparison can be seen in Figure 8 (b). It appears that the result for changing number of hidden nodes in a single layer is much more affected by number of trainable parameters than the result for varying number of hidden layers. Due to difficulties in training with a higher number of hidden layers, the number of patterns could not be reduced to 50 for the results in Figure 8 (a) to have acceptable validation performances, which makes the conditions for the models compared in figure 8 (b) differ.

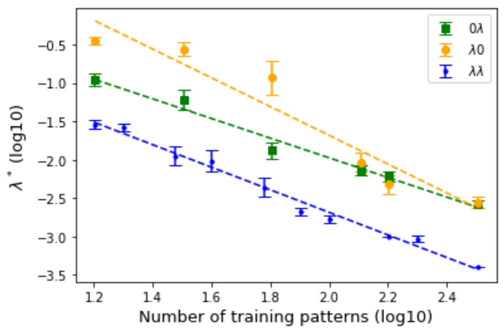
4.4 Varying λ over layers



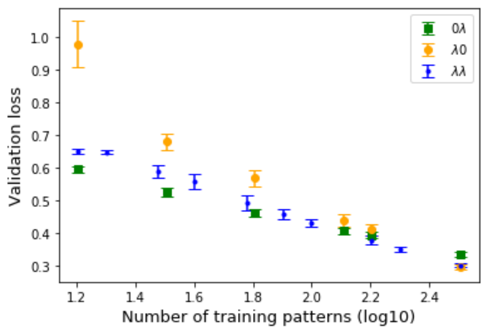
(a) λ^* in the second layer over λ in the first layer



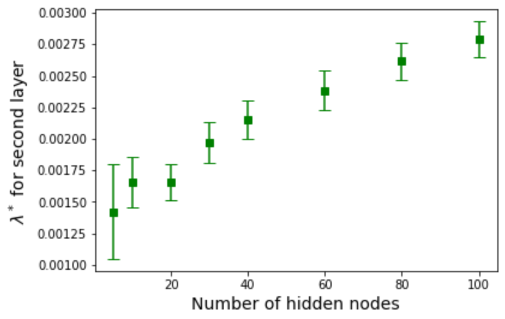
(b) Validation performance for (a)



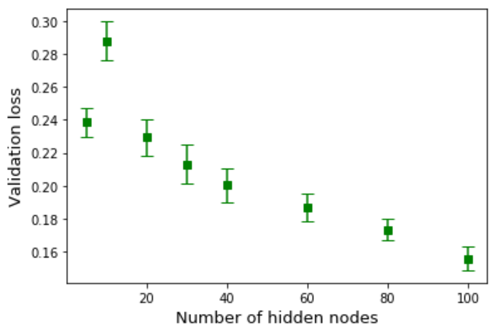
(c) λ^* over number of training patterns for λ only in the second layer (0λ), first layer ($\lambda 0$), or with the same value in both layers ($\lambda\lambda$)



(d) Validation performance for (c)



(e) λ^* in the second layer over number of hidden nodes, with $\lambda = 1e\text{-}5$ in the first layer



(f) Validation performance for (e)

Figure 9: λ^* (a) varied over layers and (c) only applied to one layer or with the same value to both, over number of training patterns. The plots in (a) and (e) were produced using 5000 epochs, 100 training patterns and different numbers of hidden nodes in one layer. The plots in (b) were produced using 5000 epochs and 20 hidden nodes in one layer.

It appears that the validation performance is better and λ^* values agree more with the hypothesis for small values on λ in the first layer and a freely chosen λ in the second layer. There also appears to be a clear relation between size of hidden layer and λ^* in the second layer, for the data points with best validation performance. A linear regression on the last six data points of Figure 9 (e) in a log-log plot gives a slope of $+0.31$. This is still far from the hypothesis proposed $+1$ relation, although a much more promising relation than what was found in Figure 7 (b) for 100 patterns, where λ was applied to all layers.

5 Discussion

Number of training patterns
A $-\log(N)$ trend for $\log(\lambda^*)$ could be seen in Figure 5 for this study, which agrees with the hypothesis. Even so, Figure 6 (a) shows that the slope depended on number of hidden nodes and was less clear for a smaller interval of pattern numbers. Formally, since the steps in the range for λ were discrete, the exact best λ values are unlikely to have been found. However, the range for λ in Table 3 appears to have been appropriate in terms of assessing the hypothesis, and taking the mean of nine samples for every study helped compensate for this step size. The study of λ^*-dependence on number of training patterns was additionally complemented by the use of another data set, D2, where the effects of having different fixed numbers of training patterns on λ^* while studying other parameters could be observed.

Number of hidden nodes
The hypothesis predicts a linear relationship between λ_i^* for node i and number of inputs to the node. For these studies, MLPs with one hidden layer were used, so λ was applied to the input layer and one hidden layer in total. As λ_i was the same for all layers until the studies in section 4.4, it meant that λ^* would effectively be affected by the number of inputs from two layers. Therefore, a linear relation was not expected here.

Looking at the plots in Figure 7, it is difficult to decide the type of relation. The plots for 50 patterns, which required larger values of λ, showed an approximate square root dependency; but this is unclear given the size of the error bars. It can at least be said that there is some increasing trend for settings where λ is required to be greater than 0.01.

Even though 10 patterns for the same study gives an apparent increasing trend close to $+1$, the validation performance reveals that the network is practically guessing. This can be seen as the model with 10 patterns consistently produces a validation loss of $\ln(2)$, which equates to guessing for the case of balanced classes in the data set. Such a behaviour could be the result of all weights but the biases being driven to zero, and the model acting using only the bias node from the hidden layer.

A more clear monotonically increasing trend was found from Figure 9 (e), when λ^* was chosen independently for the second layer. Although the need for L2 regularization was very small, as for the same settings with λ applied to all layers, the positive trend was much greater, at +0.31 compared to +0.035. This suggests a stronger dependence on number of inputs from previous layer when λ_i is chosen independently. As the number of input dimensions was not changed in any of these studies, choosing λ^* independently for the first layer was not investigated thoroughly enough to be included in the results.

Number of hidden layers

There appeared to be a relation between hidden layers and λ^* when varying number of training patterns, such that a higher number of hidden layers produced a higher value for λ^*. Such a relation was not predicted by the hypothesis, but is reasonable, as more layers introduce more trainable parameters and a more flexible model, resulting in a greater need for L2 regularization. Interestingly, λ^* was much less dependent on number of trainable parameters when layers were added than when nodes were added to a single layer, in some agreement with the hypothesis.

It was difficult to produce clear results from the study of relation between λ^* and number of hidden layers while keeping other parameters fixed. Having comparable settings was prioritized, and only number of hidden nodes per layer and training patterns were varied. As a consequence, several results produced during this limited model selection could not be used due to improper training or lacking need for L2 regularization.

Varying λ over layers

It was found that the best validation performance was achieved for low values of λ on the first layer, and moderate values on the second. A tendency to select higher λ^* values for higher number of hidden nodes was also seen, which is reasonable as λ^* here was for the second layer, whose number of inputs is given by number of hidden nodes. Figure 9 (d) shows that the validation performance was found to be more consistent and better in general for the case where λ was only applied to the second layer than only the first.

Additionally, the data points for these settings followed the slope of their linear regression in the log-log plot much more closely, showing a greater tendency for a relation. This model even performed better than when the same λ was applied to both layers for below 80 training patterns. Also, it consistently required bigger values for λ^*, which is expected as the weights only have one layer to be regularized in. Other regularization methods applied to certain layers, such as dropout, might show similar behaviour where it is better applied to the second layer, and not the first, of an MLP with a single hidden layer.

Future studies could be made, varying which layers L2 regularization is applied to, for different numbers of hidden layers.

6 Conclusion

Dependence of best L2 strength λ^* on number of training patterns and hidden nodes was studied. The relation between λ^* and number of training patterns, being approximately inverse for some model settings, agreed more with the hypothesis than the relation for number of hidden nodes. The trend for hidden nodes was clearer for λ^* in the second layer, when λ was chosen independently over layers, in some agreement with the hypothesis. The trend was however not linear.

No clear correlation between λ^* and number of hidden layers could be found. This study was also impeded by difficulties of training deeper ANNs. It appeared that λ^* has a similar relation to number of training patterns when only applied to the single hidden layer and not input layer. A higher validation performance was observed for this model compared to the one where λ was applied to both layers, for a certain range of training patterns. Finally, the best performance was achieved when λ^* could be chosen freely for both input and hidden layer.

References

[1] P. Edén, M. Ohlsson, Introduction to Artificial Neural Networks and Deep Learning, 2021.

[2] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, Journal of Machine Learning Research, vol. 13, no. Feb., p. 282, 2012.

[3] X. Glorot, Y. Bengio, Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, PMLR 9:249-256, 2010.

[4] D. Rumelhart, G. Hinton, R. Williams, Learning representations by back-propagating errors. Nature 323, p. 533–536. 1986.

[5] T. Bayes, An essay towards solving a problem in the doctrine of chances. Phil. Trans. of the Royal Soc. of London, 53, 370–418. 1763.

[6] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, 15(56):19291958, 2014.

[7] F. Chollet & others, 2015. Keras. Available at: https://github.com/fchollet/keras