

https://dspace.org/

Generalizing regularization of neural networks to correlated parameter spaces

Jarvis, Devon

2021
https://hdl.handle.net/10539/33075

Downloaded from a DSpace repository. https://dspace.org/

Generalizing Regularization of Neural Networks to Correlated Parameter Spaces

Master of Science Research Dissertation

School of Computer Science & Applied Mathematics
University of the Witwatersrand

Devon Jarvis
1365149

Supervised by Dr. Richard Klein, Prof. Benjamin Rosman and Prof. Andrew Saxe

September 10, 2021



A dissertation submitted to the Faculty of Science, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science

Abstract

A common assumption of regularization techniques used with artificial neural networks is that their parameters are independently distributed. This is primarily done for simplicity or to enforce this constraint on the model parameters. In this work we provide theoretical and empirical results showing that the independence assumption is unreasonable and unhelpful for regularization. We create and evaluate a novel regularization method called Metric regularization which adapts the degree of regularization for each parameter of the network based on how important the parameter is for reducing the loss on the training data. Importantly Metric regularization accounts for the impact that a parameter has on the other model parameters to determine how important it is for reducing the loss. Thus, our novel regularization method adapts to the correlation of the parameters in the model. We provide theoretical results showing that Metric regularization has the Minimum Mean Squared Error property. We also evaluate the utility of Metric regularization empirically and find that it is damaging to the model which is unable to effectively fit the training data as a result. We instead find that regularization methods which adaptively choose to regularize only the parameters which are unhelpful for fitting the training data are able to improve the generalizability of the networks without hindering the training data performance. We provide justifications for the apparent disconnect between our theoretical and empirical results for Metric regularization and in so doing shed some light on what causes a generalization gap with networks as well as the impacts of different initialization regimes used when training networks.

Declaration

I, Devon Jarvis, hereby declare the contents of this research dissertation to be my own work. This dissertation is submitted for the degree of Master of Science in Computer Science at the University of the Witwatersrand. This work has not been submitted to any other university, or for any other degree.

Devon Jarvis
10 September 2021

Contents

Abstract i

Table of Contents iv

1 Introduction 1

2 Background and Related Work 7
- 2.1 Introduction . . . 7
- 2.2 Objective Bayesian Statistics and the Jeffreys Prior . . . 7
- 2.3 Gradient Descent and Regularization Methods . . . 11
- 2.4 Riemannian Geometry . . . 14

3 Theoretical Analysis 19
- 3.1 Introduction . . . 19
- 3.2 Parameters Along a Path are Correlated . . . 20
- 3.3 Effect of Cramér-Rao Lower Bound on Gaussian Distribution . . . 22
- 3.4 Fisher Information of Gaussian Likelihood and Posterior Distributions . . . 23
- 3.5 Derivation of Metric Regularizer from Bayesian Prior . . . 24
- 3.6 Equivalence to MMSE for Linear Regression . . . 26
- 3.7 Equivalence to Maximum A Posteriori . . . 28
- 3.8 Computing Multiplication of Hessian by a Vector . . . 29
- 3.9 Simultaneous Power Method using Hessian Multiplication . . . 30
- 3.10 Conclusion . . . 32

4 Experimental Methodology 33
- 4.1 Introduction . . . 33
- 4.2 Datasets . . . 33
  - 4.2.1 XOR and XORD Datasets . . . 33
  - 4.2.2 MNIST Datasets . . . 35
  - 4.2.3 Synthetic Dataset . . . 35
  - 4.2.4 CIFAR-10 Dataset . . . 36
- 4.3 Learning Rules . . . 36
- 4.4 Experimental Setup . . . 39

5 Experimental Results **46**
5.1 Introduction 46
5.2 Checking Efficient Power Method 46
5.3 XOR Datasets 48
5.4 MNIST Datasets 57
5.5 Synthetic Dataset 64
5.6 CIFAR-10 Dataset 68
5.7 Eigenvalues from Hessian of Loss 72
5.8 Conclusion 78

6 Conclusion **83**

References **90**

Chapter 1

Introduction

Artificial Neural Networks (henceforth called NNs or networks) are mathematical structures comprised of many small units or neurons, each of which perform a computation on the sum of their input data and provide a single output value. These neurons are grouped into layers with the output of one layer providing the input to the following layer, while neurons in the same layer are not connected. Connections between neurons are given weightings, which are known as the parameters of the model. The output of a neuron is multiplied by such a parameter before being passed to the following neuron. It is the goal of training to learn the parameter values that produce a target output value from the model given a set of input values. In the supervised learning framework, which we study in this work by looking at both classification and regression tasks with fully-connected feedforward network, the input-output value pairs provide examples to the model of the desired mapping which we want it to learn. The performance of the model is quantified by a loss function that measures the discrepancy between the model's output and the target output. NNs form increasingly abstract representations of the input data within their hidden layers, with the final layer using these composite representations to compute the output which minimizes the loss function for the given task.

The loss function forms a manifold over the parameter space Θ of an NN by assigning a loss value to each possible parametrization of the model $\theta \in \Theta$. This manifold is referred to as a landscape with the aim of training being to find the minimum of this landscape. The fact that a single parameter's value can only be deemed good or bad (as measured by the loss function) in the context of the rest of the model has resulted in the act of directly finding the optimal values for all parameters simultaneously being a computationally intractable task. As a result models are trained through iterative optimization methods, such as Gradient Descent (GD). These methods, however, are prone to falling into sub-optimal local minima within the loss landscape, take long to develop a competent model and are sensitive to factors such as how the model parameters were initialized. The fact that a wealth of data is required to train an NN is a further hindrance. In addition the data used to train the model must be representative of the entire population of data from which it is sampled. In many cases obtaining a fully representative sample of enough training examples is infeasible, resulting in

the model having to use learned patterns from the data that it has seen to try and generalize to the unseen data. In essence the problem of developing a model that generalizes well is the problem that NNs, as well as other machine learning techniques and models, aim to solve, with the problem of fitting the training data being trivial given a large enough model formed out of non-linear composite functions.

There have been many adaptations to GD that aim to speed up training or improve the generalization of trained networks. A large majority of these adaptations aim to utilize additional pre-determined values that bias the model parameters towards certain spaces within the loss landscape. These methods are known as regularizers and the pre-determined values are hyper-parameters that are set independently of the model parameters. Another common approach is to utilize the shape of the loss landscape in the area of the model parameters to take more precise updates at each time step. All of these adaptions have been successfully implemented in a variety of situations (for example L2 regularization [Krogh and Hertz 1992]), however, none have proven to work consistently. This fact, along with the fact that we are still learning about the relationship between the loss landscape and how networks train, means that the best method to train and regularize a model remains an open question within the field of machine learning. We, thus, explore this relationship further by proposing and investigating a novel regularization method defined by: $\theta^* = argmin_\theta(L(\mathbf{X}, \theta) + \theta^T I(\theta)\theta)$. Here $L(\mathbf{X}, \theta)$ is any loss function dependent on the parameter values θ, $\mathbf{X}$ is the training data set, $I(\theta)$ is the Fisher Information Matrix (which we provide background on in Section 2.2) and the second term in the $argmin_\theta$ is the novel regularizer. This method is aimed at removing the independence assumption between neurons that is used with prior regularization methods (such as L1 and L2 regularization) by accounting for the significance of each parameter of the model as a whole when regularizing. We refer to this method as Metric regularization. In this work we refer to the "independence assumption" in relation to methods which do not account for the interaction between parameters or treat parameters separately when regularizing (as L1 and L2 regularization do). This is similar in a Bayesian sense to assuming parameters are uncorrelated, indeed this is the case for deriving L1 and L2 regularization as we show in Chapter 2. However, our definition of independence is broader as we mean for this terminology to also incorporate methods which are agnostic to the correlation between parameters.

To gain some initial intuition for this regularizer, and what it means to remove the independence assumption, we plot the vector field from the Metric and L2 regularizers in Figures 1.1a and 1.1b respectively for a two-parameter network shown in Figure 1.2. From these plots we see that L2 forces the parameters towards the origin, and a parameter is always pushed towards 0 regardless of the other parameter value. In contrast, Metric regularization follows the shape of the landscape and does not always point towards the origin. In particular when one parameter has a large value relative to the other, it is regularized very little. We then see that, for this particular landscape, Metric regularization promotes sparsity in the networks and L2 regularization promotes a low-norm solution. We believe this property of Metric

regularization to adapt to the relative values of the parameters to be valuable, as the importance of one parameter value can only be determined in the context of the rest of the network.

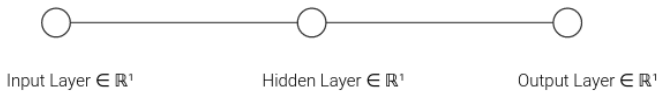
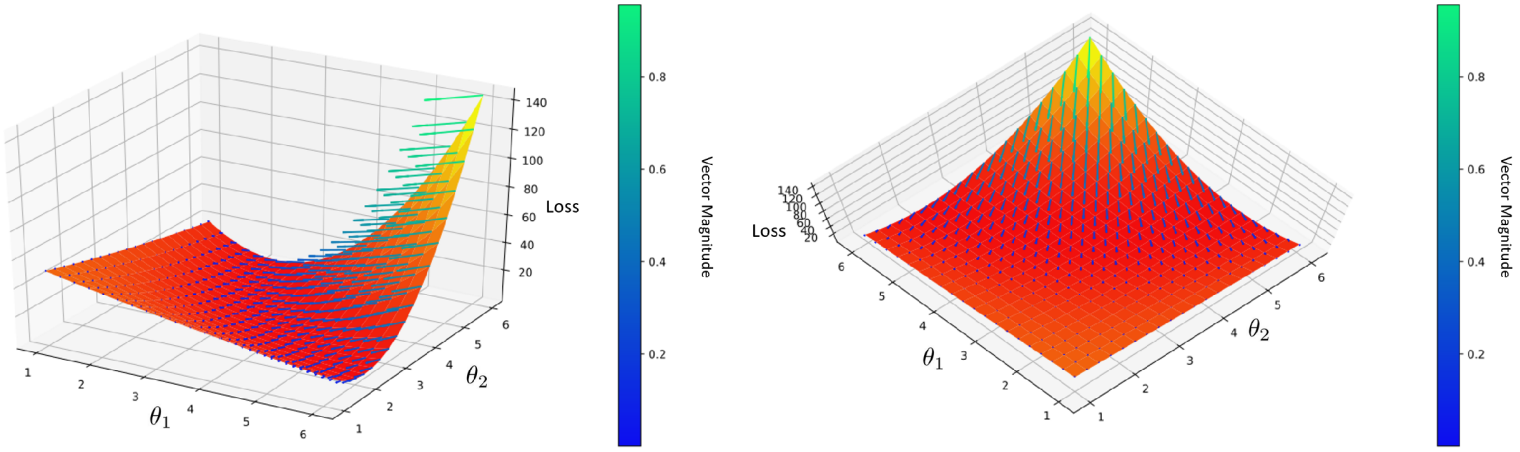


Figure 1.2: NN with a single path and 2 parameters

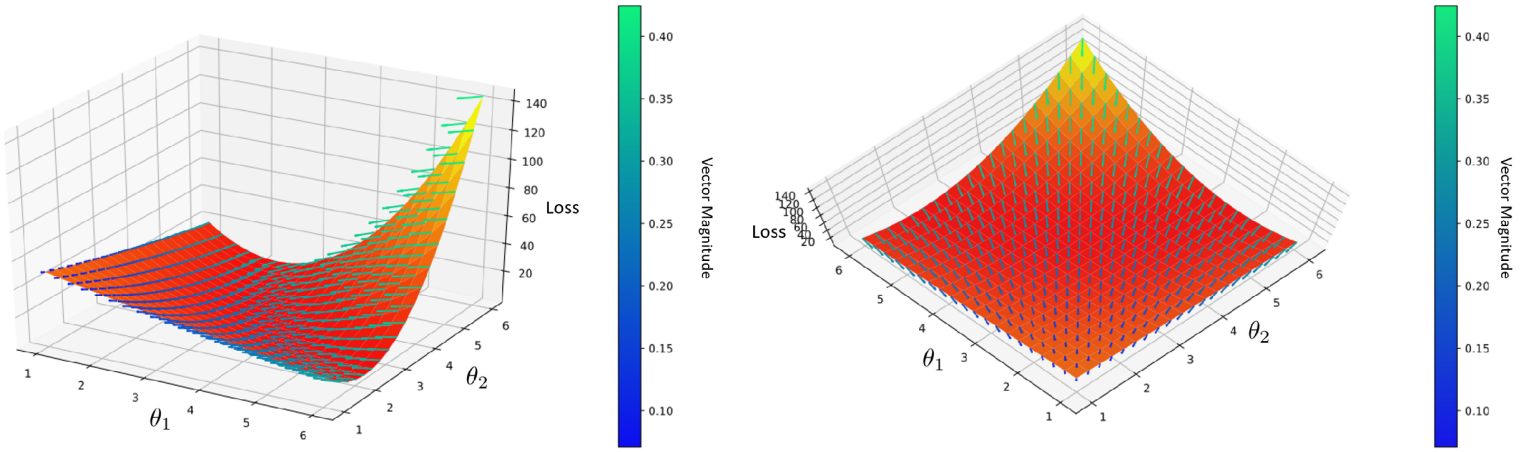
Due to prior work, such as the bias-variance trade-off [Geman *et al.* 1992] and Minimum Description Length Principle [Rissanen 1978], there is a common notion that there is a trade-off between NNs fitting the training data and generalizing to unseen data. This is also shown in more recent works referencing Energy-Entropy competition where energy references the loss of the model and entropy references the variance in its parameters [Zhang *et al.* 2018]. In this work we explore this trade-off by adaptively increasing the regularization of parameters that are more significant to the model fitting the training data. Through this we aim to see whether increasing the regularization of these significant parameters improves the generalization of the model. In terms of the bias-variance trade-off we more severely restrict the variance of the parameters which are most useful for decreasing the bias of the model. Ultimately we find compelling results in favour of approaches contrary to Metric regularization which adaptively choose to only regularize insignificant parameters. In so doing we also reflect the absence of a trade-off between minimizing training loss and generalizing to unseen data. Practically, this suggests a revision of our beliefs around overfitting and the causes of generalization gaps in NNs.

This dissertation proceeds as follows. In Chapter 2 we provide information on the necessary prior and related work for our research. This chapter is comprised of the following sections. Section 2.2 discusses the necessary background on the Jeffreys prior and Objective Bayesian statistics, and introduces the Leibniz integral rule that we use multiple times throughout this work. Section 2.3 then discusses the current state of the art regularization methods used in machine learning, while Section 2.4 provides the necessary background on Riemannian Geometry. We also briefly discuss recent work on the spectrum of the Hessian matrix of the loss of NNs. A vital concept discussed in this section and introduced in Section 2.2 is the Fisher Information Matrix and its use as a metric tensor for Riemannian manifolds.

We then proceed with Chapter 3 where we present the theoretical results of this work, motivating the use of the Metric regularization method. We begin with two preliminary results in Section 3.2 and Section 3.3 where we show that the parameters of an NN are correlated and the effect of using the Cramér-Rao bound on a Multivariate Gaussian distribution. In Sections 3.5, 3.6 and 3.7 we then present the three parts of the larger proof which reflects that the use of a regularizer derived from the Multivariate Gaussian distribution (Section 3.5) results in model parameters equivalent to the minimum mean squared error parameters



(a) Metric Regularizer Vector Plot



(b) L2 Regularizer Vector Plot

Figure 1.1: Vector plots showing the different regularization steps from Metric regularization in Figure 1.1a and L2 regularization in Figure 1.1b. Vector plots are shown from two angle, the front (left images) and top (right images) view. Each point on the θ_1 and θ_2 axis is parametrization for the network in Figure 1.2 and map to a corresponding loss value (we generated data from an identical "teacher" network). We see that L2 regularization always points towards the origin, while Metric regularization follows the slope of the data (note at the corners of the plot the vectors run parallel to one of the parameters which has little effect on the network's function).

for linear regression models (Section 3.6) as well as the same parameters found from Bayesian inference (Section 3.7). We conclude this chapter with two related sections, Section 3.8 and 3.9, where we present the theory behind an algorithm for finding the spectrum of the Hessian of an NN in a memory efficient manner.

In Chapter 4 we then describe our experimental methodology used to evaluate the Metric regularizer. In Section 4.2 we present the five datasets used, namely the XOR, XORD, MNIST, a synthetic dataset and the CIFAR-10 datasets. In Section 4.3 we present the various learning rules used in the experiments. Naturally these learning rules include vanilla Stochastic Gradient Descent (SGD) as well as the learning rule with the Metric regularizer, however, we also include various baseline regularizers and other regularizers used to compare with. The chapter concludes with Section 4.4 where we describe the setup of the experiments and the considerations around this setup.

Throughout this work we specify the notation used for an equation along with the given equation. We do adhere to some notational standards. Bold font is used to denote a vector ($\mathbf{x}$) with $\mathbf{0}$ being a vector of 0's. Capital letters are used to denote random variables (X) and capital letters with indices denote elements of matrix variables (for example X_{ij} shows the i-th row and j-th column of the matrix X). $\sum$ is used to denote the summation operator, while $P(X = x)$ shows the probability of random variable X taking the value x. We abbreviate this to $P(X)$ to lighten notation. We utilize $\log()$ to denote the natural logarithm, while all other log bases are explicitly indicated, for example $\log_{10}()$. Finally $I(\theta)$ denotes the Fisher Information Matrix (with the dependence on θ) and $\mathbf{I}$ denotes the identity matrix. N is the number of parameters in our model and P is the number of data points in our training data set. We also use hats on variables to denote that they are estimates of variables for example $\hat{y}$.

As a notational and practical standard of this work we apply a learning rate (α) to the derivative of the loss and a regularization rate (λ) to the derivative of the regularization term. We treat these two components individually. In some other works the learning rate is applied to both the loss and regularization terms. We separate these two components for readability and to make hyper-parameter tuning simpler. If the latter version is preferred then in our work the notation can be seen as $\lambda = \lambda'/\alpha$ where λ' is the regularization rate and λ is the ratio of the regularization to learning rates.

There are five primary contributions of this work

- We create a set of new regularizers which adapt the regularization of a parameter to how important the parameter is to fitting training data.
- We present theoretical analysis on why the new regularizers are necessary and in particular on the benefits of Metric regularization.
- We make all of our novel regularizers efficient, which we define as the regularizer being

of the same time and memory complexity as GD. Using the same technique we use to make our regularizers efficient we also create a novel algorithm for finding the top Eigenvalues of the Hessian of the loss of an NN.

- We show insight into the relationship between NNs fitting the training data and generalizing to unseen data. We find that there is not a trade-off between training and test data performance.
- We begin to bridge the gap between the small and large parameter initialization regimes [Geiger *et al.* 2020] and show the implicit regularization effect of using small initial parameters.

Chapter 2

Background and Related Work

2.1 Introduction

In this chapter we provide the background necessary for this research. We begin with Section 2.2 in which we describe Objective Bayesian statistics and in particular the Jeffreys prior which is useful for the theoretical results in Chapter 3. In particular we use the Jeffreys prior defined in this section to relate the Maximum A Posteriori (MAP) parameters to the Bayesian parameters for an NN. This is used to motivate the use of Metric regularization. This is then followed by Section 2.3 in which we discuss some of the current regularization methods employed when training neural networks (NNs). We focus primarily on regularizers which are derived from Bayesian priors as these are the most similar to the Metric regularization methods. In particular we compare metric regularization against L2 regularization throughout this work and use it in contrast to our new regularizers to interpret the performance of models on unseen data. Through this we gain insight into the factors causing the generalization gap between training and test data performance. We additionally, briefly, discuss Dropout which is a particularly popular regularization method that is Bayesian in nature, but not explicitly derived from a prior. Finally in Section 2.4 we provide a brief background on the Riemannian Geometry perspective of the loss landscape of an NN. In particular we discuss the Fisher Information Matrix and its use as a metric tensor. The metric tensor is the main component of the metric regularizer and so we provide a lot of detail on its utility and use in Riemannian geometry. Our ability to generalize regularization to correlated parameter spaces is due to the relationship between the Fisher Information Matrix and covariance matrix which we also show in this section.

2.2 Objective Bayesian Statistics and the Jeffreys Prior

In this section we look at Objective Bayesian statistics, a sub-field of Bayesian statistics which focuses on imparting as little information on the parameter inference as possible. In other words, it aims to ensure that the data has the most influence possible over the inference. This is useful if we know little about the parameter values prior to observing any

data, which is likely the case for the parameters of NNs. Naturally we begin by defining Bayes' rule [Bayes 1763] which provides a formula for including our prior knowledge into the process of modelling data. Through this formula we may determine the probability of a certain parameter value, given a set of data and our own prior beliefs on the parameter's value. We then move on to Objective Bayesian statistics. Bayes' rule is defined as:

$$P(\theta|\mathbf{X}) = \frac{P(\mathbf{X}|\theta)P(\theta)}{\int P(\mathbf{X}|\theta)P(\theta)d\theta} = \frac{1}{Z^*}P(\mathbf{X}|\theta)P(\theta) \tag{2.1}$$

$P(\mathbf{X}|\theta)$ is known as the likelihood distribution and reflects the probability of a batch of data $\mathbf{X}$ occurring under a probability distribution parametrized by θ. In general we have to make an assumption (based on the data) as to the format of the likelihood distribution $P(\mathbf{X}|\theta)$. For example if we know that the data only takes on the values $\{0, 1\}$ then we may choose to model this data with a Bernoulli distribution, which models data with only two possible discrete values. $P(\theta)$ in Equation 2.1 is known as the prior distribution over the parameters of the likelihood distribution. It is defined over Θ, the parameter space of the distribution, and provides a probability to each $\theta \in \Theta$. The support of the prior distribution is determined by the choice of likelihood distribution. Finally, the denominator of Equation 2.1 is known as the evidence and reflects the probability of the data $\mathbf{X}$ under the entire parameter space Θ. In other words, the probability of $\mathbf{X}$ independent of the choice of the value assigned to θ. The denominator ensures the resultant distribution formed in Equation 2.1 integrates to 1 and is a proper distribution. The denominator is also denoted by Z^* a normalizing constant. The use of the superscript star on Z^* is to reflect that it is the normalizing constant resulting from the likelihood and prior distribution. This is to distinguish from a normalizing constant for the likelihood distribution only, Z, which is used later in this work. Bayes' rule results in the posterior distribution $P(\theta|\mathbf{X})$ which provides the probability of a model parametrization given a fixed set of data. Similar to the prior distribution, the posterior distribution is defined over the parameter space of the model and can be seen as the prior distribution having been updated after observing the values of the data set.

The goal of solving parameter inference problems is to determine the value for θ which maximizes this posterior probability given a fixed set of data. Formally we aim to find, $\text{argmax}_{\theta\in\Theta} P(\theta|\mathbf{X})$. For this work we are concerned with how we set the prior distribution and its effects on the posterior distribution. One of the most common distributions to use for the prior is the uniform distribution $P(\theta) \propto 1$ in which a constant value is provided for all $\theta \in \Theta$. This results in $P(\theta|\mathbf{X}) \propto P(\mathbf{X}|\theta)$ and so the value which maximizes $P(\theta|\mathbf{X})$ is the one which maximizes the probability of the data set under the likelihood distribution $P(\mathbf{X}|\theta)$. This is known as Maximum Likelihood (ML) approximation. In many cases, however, it is advisable to use a non-uniform prior distribution. In this case the prior distribution changes the resultant posterior distribution and we say it "informed" the posterior distribution. We show two examples of such priors in Section 2.3 below. When we use a non-uniform prior and find the parameter value which maximizes the posterior probability distribution we are using Maximum A Posteriori (MAP) approximation.

A difficulty associated with setting the prior for use with Bayes' rule is that it is sensitive to the choice of model parameters. Thus, a prior which is uninformative in one parametrization may be very informative in another, after a change of variables. This means that, given the same data and assumption on the form of the likelihood distribution, two different posterior distributions may occur depending on the parametrization used. This breaks consistency, a frequentist notion that an inference should converge to a consistent answer regardless of the parametrization. This lead to the creation of the Jeffreys prior [Jeffreys 1946]. The Jeffreys prior is defined as:

$$p(\theta) \propto \sqrt{\det I(\theta)} \tag{2.2}$$

where $I(\theta)$ is the Fisher Information matrix [Fisher 1922] defined (where the expectation is over the dataset) as:

$$[I(\theta)]_{ij} = E_{\mathbf{X}}\left[\left(\frac{\partial}{\partial\theta_i}\log P(\mathbf{X}|\theta)\right)\left(\frac{\partial}{\partial\theta_j}\log P(\mathbf{X}|\theta)\right)|\theta\right] \tag{2.3}$$

Fisher Information Matrix can be viewed as the covariance of gradients from the log-likelihood of a probability distribution. Intuitively it reflects how much influence a parameter has over the other parameters and the probability distribution as a whole. The determinant of the Fisher Information Matrix then shows how sharply the probability distribution changes in all directions. In other words, it shows the total variance, away from 0, in the gradients of the parameter values of the log-likelihood. The Jeffreys prior then adjusts its prior probability based on this variance in the gradients. Lower variance reflects that a parameter is less important to the overall distribution, and so we are less confident that it has been accurately restricted by the data.

The first important aspect of the Jeffreys prior is the fact that it is invariant under a change of variables (a reparametrization of a statistical model) and provides the same relative probability density to a region of parameter space regardless of the original parametrization of a model. In other words, whether the modelling process is conducted in θ space directly or in some other parameter space $\phi = f(\theta)$ first and then transformed to θ space the resultant prior density will be the same. The same cannot be said for other prior densities in general. We demonstrate this property in the one-parameter case, with $\theta \in \mathbb{R}$ and $\phi \in \mathbb{R}$. Let $p(\theta) \propto \sqrt{I(\theta)}$ where the determinant is dropped in the one dimensional case. Additionally let $p(\phi) \propto \sqrt{I(\phi)}$. Then using the change of variables theorem [Kaplan 1952]:

$$p(\theta) = p(\phi)\left|\tfrac{d\phi}{d\theta}\right| \propto \sqrt{I(\phi)}\left|\tfrac{d\phi}{d\theta}\right| = \sqrt{I(\phi)\left(\tfrac{d\phi}{d\theta}\right)^2} = \sqrt{E\left[\left(\tfrac{d\log P(\mathbf{X}|\phi)}{d\phi}\right)^2\right]\left(\tfrac{d\phi}{d\theta}\right)^2}$$
$$= \sqrt{E\left[\left(\tfrac{d\log P(\mathbf{X}|\phi)}{d\phi}\tfrac{d\phi}{d\theta}\right)^2\right]} = \sqrt{E\left[\left(\tfrac{d\log P(\mathbf{X}|\theta)}{d\theta}\right)^2\right]} = \sqrt{I(\theta)}$$

The invariance property of the Jeffreys prior is due to the determinant of the Fisher Information Matrix being invariant under a change of variables [Ly *et al.* 2017]. This is the case

as the Fisher Information Matrix is the metric tensor of a statistical manifold, and since volume is an invariant property of a manifold, the determinant of the metric tensor is invariant. The Fisher Information and Riemannian manifolds are discussed further in Section 2.4. The primary utility of the Jeffreys prior, however, is in its use as an uninformative (objective) prior as it aims to impart as little information on the posterior distribution as possible [Jaynes 1968 2003]. This is particularly clear in the case of a Gaussian distribution with fixed variance (σ^2). The Jeffreys prior for the mean in this case is $P(\mu) \propto \frac{1}{\sigma}$, a uniform distribution. Thus, after any reparametrization of the mean of a Gaussian, the Jeffreys prior is equivalent to a uniform distribution in the original μ space. Thus, the Jeffreys prior is always equivalent to a uniform distribution in some parameter space, but not necessarily the parameter space which we are working in. In the case of an NN, the Jeffreys prior is uniform over the loss landscape (if we treat the output of the network as a random variable) but will not be uniform in Θ-space. In general the Jeffreys prior aims to provide less prior density to high variance areas of the parameter space. This results in a prior which places less density over a portion of the parameter space when a large set of parametrizations are likely. Likewise, portions of the parameter space where the parametrization is certain, or few parametrizations are equally likely to produce the data, obtain a high prior density. Thus, the Jeffreys prior matches the uncertainty of a parametrization and as a result informs the posterior distribution as little as possible.

Finally, it is helpful to note the relationship between the Fisher Information Matrix and Hessian of the negative log-likelihood of a probability distribution $\log(P(\mathbf{X}|\theta))$ (Section 2.3 shows that the negative log-likelihood of a Gaussian distribution results in the quadratic loss function). At the point of the maximum likelihood estimate the expected Fisher Information Matrix and the expected Hessian are equal:

$E_{\mathbf{X}}[H[-\log P(\mathbf{X}|\theta)]] = E_{\mathbf{X}}[-\frac{\partial}{\partial\theta_i}\frac{\partial}{\partial\theta_j}\log P(\mathbf{X}|\theta)] = E_{\mathbf{X}}\left[-\frac{\partial}{\partial\theta_i}\left(\frac{\frac{\partial}{\partial\theta_j}P(\mathbf{X}|\theta)}{P(\mathbf{X}|\theta)}\right)\right]$

$= E_{\mathbf{X}}\left[-\frac{(\frac{\partial}{\partial\theta_i}\frac{\partial}{\partial\theta_j}P(\mathbf{X}|\theta))P(\mathbf{X}|\theta)}{P(\mathbf{X}|\theta)^2} + \frac{\frac{\partial}{\partial\theta_i}P(\mathbf{X}|\theta)\frac{\partial}{\partial\theta_j}P(\mathbf{X}|\theta)}{P(\mathbf{X}|\theta)^2}\right] = -E_{\mathbf{X}}\left[\frac{\frac{\partial}{\partial\theta_i}\frac{\partial}{\partial\theta_j}P(\mathbf{X}|\theta)}{P(\mathbf{X}|\theta)}\right] + E_{\mathbf{X}}\left[\frac{\frac{\partial}{\partial\theta_i}P(\mathbf{X}|\theta)}{P(\mathbf{X}|\theta)}\frac{\frac{\partial}{\partial\theta_j}P(\mathbf{X}|\theta)}{P(\mathbf{X}|\theta)}\right]$

$= E_{\mathbf{X}}\left[\left(\frac{\partial}{\partial\theta_i}\log P(\mathbf{X}|\theta)\right)\left(\frac{\partial}{\partial\theta_j}\log P(\mathbf{X}|\theta)\right)\right] = [I(\theta)]_{ij}$ for $i, j \in \{1, ..., N\}$. The equality above relied on the fact that at the maximum likelihood estimator the term $E_{\mathbf{X}}\left[\frac{\frac{\partial}{\partial\theta_i}\frac{\partial}{\partial\theta_j}P(\mathbf{X}|\theta)}{P(\mathbf{X}|\theta)}\right] = 0$. This can be shown using the special case of the Leibniz integral rule where the limits of the integral are constant (which may be sufficiently large to be considered infinite):

$$\int_{-\infty}^{\infty}\frac{\partial}{\partial\theta}f(x,\theta)dx = \frac{\partial}{\partial\theta}\int_{-\infty}^{\infty}f(x,\theta)dx \tag{2.4}$$

Using Equation 2.4 we can change the order of integration and differentiation to obtain:

$E_{\mathbf{X}}\left[\frac{\frac{\partial}{\partial\theta_i}\frac{\partial}{\partial\theta_j}P(\mathbf{X}|\theta)}{P(\mathbf{X}|\theta)}\right] = \int_{-\infty}^{\infty}\frac{\frac{\partial}{\partial\theta_i}\frac{\partial}{\partial\theta_j}P(\mathbf{X}|\theta)}{P(\mathbf{X}|\theta)}P(\mathbf{X}|\theta)dx = \int_{-\infty}^{\infty}\frac{\partial}{\partial\theta_i}\frac{\partial}{\partial\theta_j}P(\mathbf{X}|\theta)dx$

$= \frac{\partial}{\partial\theta_i}\frac{\partial}{\partial\theta_j}\int_{-\infty}^{\infty}P(\mathbf{X}|\theta)dx = \frac{\partial}{\partial\theta_i}\frac{\partial}{\partial\theta_j}1 = 0.$

As a result the Jeffreys prior can also be written as $p(\theta) \propto \sqrt{\det H(\theta)}$ at the MLE estimate.

2.3 Gradient Descent and Regularization Methods

In this section we provide a brief background into related regularization methods to our proposed method. We first reflect the derivation of L1 and L2 regularization from priors on a probability distribution. These are the two most closely related regularizers to our own and their use reflects that the MAP parametrization for a model is found. The MAP parameters are those with the maximum posterior probability shown in Equation 2.1. They are the parameters that best describe the given data while accommodating our prior beliefs, which we control now through the use of regularizers. We then conclude this section with a brief look at the dropout regularization method which is probabilistic in nature, but is not derived as an explicit prior on a model. We begin, however, with the definition of Gradient Descent.

Let $\hat{y}_i = f(x_i, \theta)$ be a function defined by an NN where x_i is the input to the network, $\hat{y}_i$ is the output of the network ($\hat{y}_i = E_{\mathbf{X}}[y|x]$) and θ is a vector of model parameters. Then let L represent a loss function, one example being the quadratic loss function $L = \frac{1}{P}\sum_{i=1}^{P}(y_i - f(x_i, \theta))^2$. Here P is the number of data points in the training data set. Then the aim of training an NN is to find the parametrization $\theta \in \Theta$ which minimizes this loss function. This is achieved by repeatedly moving the parameter values in the negative direction to the gradient of the loss with respect to the parameters. Thus, we update the parameters with the learning rule $\theta_{i(t+1)} = \theta_{i(t)} - \alpha\frac{\partial}{\partial\theta_{i(t)}}L$ where the subscript i reflects which parameter is being updated in the parameter vector. The subscript in brackets (t) reflects the time step of the update and α is a pre-defined learning rate. We can then add a regularization term to the learning rule aimed to restrict the parametrization to certain regions of the parameter space. The learning rule takes the form $\theta_{i(t+1)} = \theta_{i(t)} - \alpha\frac{\partial}{\partial\theta_i}(L + R)$. The aim is then for the model to find the θ value corresponding to the minimum in the landscape defined by $(L + R)$, the loss function and regularization term.

In the following we derive the L1 and L2 regularizers from their corresponding probability distributions applied to Bayes' rule. For both derivations we use a one-dimensional Gaussian distribution for the likelihood distribution $P(\mathbf{X}|\theta)$. This corresponds to a regression problem using the quadratic loss. In traditional problems solved by NNs only the position parameters (θ) of this likelihood distribution are learned. As a result it is necessary to assume the scale parameter (variance) is constant. We also denote the prior distribution over θ as $P(\theta|\mu)$ where μ is a parameter of the prior distribution that we set (a hyper-parameter). L1 and L2 regularization correspond to the use of $\mu = \mathbf{0}$. Bayes' rule applied to these distributions is as follows:

$$P(\theta|\mathbf{X}, \mu) = \frac{P(\mathbf{X}|\theta)P(\theta|\mu)}{\int P(\mathbf{X}|\theta)P(\theta|\mu)d\theta} = \frac{1}{Z^*}P(\mathbf{X}|\theta)P(\theta|\mu) \tag{2.5}$$

For L1 regularization a Laplace distribution is used for the prior where b is its corresponding scale parameter, which is also assumed to be constant. We begin with this derivation, where the Laplace distribution is shown in Equation 2.6. This is then followed by L2 regularization

with a Gaussian prior distribution where $\hat{\sigma}^2$ is the constant scale parameter. The independent (isotropic) Gaussian distribution is shown in Equation 2.7. In both cases an independence assumption in the values of the parameters is used. It is also assumed that each data point $x \in \mathbf{X}$ is independent and identically distributed (i.i.d).

L1 Regularization Derivation

Let:

$$P(\mathbf{X}|\theta) = \frac{1}{Z}\exp\left(-\sum_{i=1}^{P}\frac{(y_i - f(x_i,\theta))^2}{2\sigma_i^2}\right)$$

$$P(\theta|\mu) = \prod_{j=1}^{N}\frac{1}{2b}\exp\left(-\frac{|\theta_j - \mu_j|}{b}\right) = \frac{1}{2b}\exp\left(-\sum_{j=1}^{N}\frac{|\theta_j - \mu_j|}{b}\right) \tag{2.6}$$

Where: μ is a hyper-parameter we set and enforces the constraint on θ

We have also assumed the parameters are independent for the prior and sum over the exponent to get the joint density over the parameters.

This results in a posterior distribution (using Equation 2.1):

$$P(\theta|\mathbf{X},\mu) = \frac{1}{2bZ}\exp\left[-\sum_{i=1}^{P}\frac{(y_i - f(x_i,\theta))^2}{2\sigma_i^2} - \sum_{j=1}^{N}\frac{|\theta_j - \mu_j|}{b}\right]$$

Taking the negative of the natural log of this posterior distribution we obtain:

$$(L+R) = -\log P(\theta|\mathbf{X},\mu) = \sum_{i=1}^{P}\frac{(y_i - f(x_i,\theta))^2}{2\sigma_i^2} + \sum_{j=1}^{N}\frac{|\theta_j - \mu_j|}{b} + \log(2bZ)$$

Since $\log(2bZ)$ is independent of the θ value it does not change the stable point of θ and so we can drop it from the loss function. We also set our hyper-parameter $\mu_j = 0 \;\forall\; j \in \{1, 2, ...N\}$.

This gives us the loss function:

$$(L+R) = \frac{1}{2\sigma_i^2}\sum_{i=1}^{P}(y_i - f(x_i,\theta))^2 + \sum_{j=1}^{N}\frac{1}{b}|\theta_j|$$

Since we used the negative log we aim to minimize this function with respect to the parameters.

The log is a monotonic function and so does not change the stable points of the parameter values.

We, thus, take the derivative with respect to θ to find its stable points.

$$\frac{\partial(L+R)}{\partial\theta_k} = \frac{1}{2\sigma_i^2}\frac{\partial}{\partial\theta_k}\sum_{i=1}^{P}(y_i - f(x_i,\theta))^2 + \frac{1}{b}\frac{\partial}{\partial\theta_k}\sum_{j=1}^{N}|\theta|$$

$$\frac{\partial(L+R)}{\partial\theta_k} = -\frac{1}{\sigma_i^2}\sum_{i=1}^{P}(y_i - f(x_i,\theta))(f'(x_i,\theta)) + \frac{1}{b}$$

In practice a constant is applied to the regularizer to determine the degree of regularization.

We denote this as λ. In addition we now include the learning rate α.

We then have the parameter update equation:

$$\theta_{i(t+1)} = \theta_{i(t)} - \frac{\partial(L+R)}{\partial\theta_i} = \theta_{i(t)} + \alpha' \sum_{i=1}^{P}(y_i - f(x_i,\theta))(f'(x_i,\theta)) - \lambda'$$

Where $\alpha' = \frac{\alpha}{\sigma_i^2}$ and $\lambda' = \frac{\lambda}{b}$ since σ_i^2 and b are both assumed constants with respect to θ.

L2 Regularization Derivation

Let:

$$P(\mathbf{X}|\theta) = \frac{1}{Z}\exp\left(-\sum_{i=1}^{P}\frac{(y_i - f(x_i,\theta))^2}{2\sigma_i^2}\right)$$

$$P(\theta|\mu) = \prod_{j=1}^{N}\frac{1}{\hat{Z}}\exp\left(-\frac{(\theta_j-\mu_j)^2}{2\hat{\sigma}^2}\right) = \frac{1}{\hat{Z}}\exp\left(-\sum_{j=1}^{N}\frac{(\theta_j-\mu_j)^2}{2\hat{\sigma}^2}\right) \tag{2.7}$$

Where: μ is a hyper-parameter we set and enforces the constraint on θ

We have also assumed the parameters are independent for the prior and sum over the exponent to get the joint density over the parameters.

This results in a posterior distribution (using Equation 2.1):

$$P(\theta|\mathbf{X},\mu) = \frac{1}{Z\hat{Z}}\exp\left[-\sum_{i=1}^{P}\frac{(y_i - f(x_i,\theta))^2}{2\sigma_i^2} - \sum_{j=1}^{N}\frac{(\theta_j-\mu_j)^2}{2\hat{\sigma}^2}\right]$$

Taking the negative of the natural log of this posterior distribution we obtain:

$$(L+R) = -\log P(\theta|\mathbf{X},\mu) = \sum_{i=1}^{P}\frac{(y_i - f(x_i,\theta))^2}{2\sigma_i^2} + \sum_{j=1}^{N}\frac{(\theta_j-\mu_j)^2}{2\hat{\sigma}^2} + \log(Z\hat{Z})$$

Since $\log(Z\hat{Z})$ is independent of the θ value it does not change the stable point of θ and so we can drop it from the loss function. We also set our hyper-parameter $\mu_j = 0 \;\forall\; j \in \{1,2,...,N\}$.

This gives us the loss function:

$$(L+R) = \frac{1}{2\sigma_i^2}\sum_{i=1}^{P}(y_i - f(x_i,\theta))^2 + \sum_{j=1}^{N}\frac{1}{2\hat{\sigma}^2}(\theta_j)^2$$

Since we used the negative log we aim to minimize this function with respect to the parameters.

The log is a monotonic function and so does not change the stable points of the parameter values.

We, thus, take the derivative with respect to θ to find its stable points.

$$\frac{\partial(L+R)}{\partial\theta_k} = \frac{1}{2\sigma_i^2}\frac{\partial}{\partial\theta_k}\sum_{i=1}^{P}(y_i - f(x_i,\theta))^2 + \frac{1}{2\hat{\sigma}^2}\frac{\partial}{\partial\theta_k}\sum_{j=1}^{N}(\theta_j)^2$$

$$\frac{\partial(L+R)}{\partial\theta_k} = -\frac{1}{\sigma_i^2}\sum_{i=1}^{P}(y_i - f(x_i,\theta))(f'(x_i,\theta)) + \frac{1}{\hat{\sigma}^2}(\theta_k)$$

In practice a constant is applied to the regularizer to determine the degree of regularization.

We denote this as λ. In addition we now include the learning rate α.

We then have the parameter update equation:

$$\theta_{i(t+1)} = \theta_{i(t)} - \frac{\partial(L+R)}{\partial\theta_i} = \theta_{i(t)} + \alpha' \sum_{i=1}^{P}(y_i - f(x_i,\theta))(f'(x_i,\theta)) - \lambda'(\theta_k)$$

Where $\alpha' = \frac{\alpha}{\sigma_i^2}$ and $\lambda' = \frac{\lambda}{\hat{\sigma}^2}$ since σ_i^2 and $\hat{\sigma}^2$ are both assumed constants with respect to θ.

Finally we look at the last regularization method, Dropout. Dropout is different from the two regularizers above as it is not derived as a prior on the parameter distribution. Dropout works by stochastically removing a certain portion of the network at each training time step. For example if $\beta = 0.5$ is the dropout rate then half of the model parameters will be set to 0 (effectively removed) for a time step of training. At any given point in time during training the capacity of the network is β of the real capacity. For the parameters that remain in the model, they are updated with any of the existing optimization algorithms. Randomly removing parameters while training with Dropout prevents parameters from adapting significantly to the state of the rest of the network and being valuable only within the context of the rest of the network. As a result every parameter of the network learns valuable information from the training data. Then when moving from training to test data, the model parameters are multiplied by β and the entire network is then used for inference on test data.

This process is often discussed in conjunction with ensemble models, where multiple versions of the same network are trained and then the average of the models' predictions is then taken for inference. Baldi and Sadowski [2013] characterize dropout as simultaneously training the ensemble of all sub-networks. Dropout's strength, however, is its speed relative to ensemble models as it does not require entirely different models to be learned, and parameter updates at one time step affect future sub-networks. The comparison also highlights a drawback of Dropout. It effectively reduces the capacity of the network and results in significant redundancy between parameters. This is because, when a parameter is dropped from the network, another parameter begins to learn the information the first parameter had already learned. Then when these model parameters are multiplied by β and both included in making inferences it is functionally the same as using the parameters of two separate networks. In addition, while Dropout may be faster than ensembles, it is still slower to train than vanilla GD. Dropout, similar to ensemble models, does provide robust parametrizations for networks and is used extensively in practice to achieve good results. While not perfect Dropout provides a practical method of obtaining ensemble-like parametrizations without the computational costs of training numerous models.

2.4 Riemannian Geometry

Given a real, smooth manifold M, at each point $p \in M$ we define a vector space T_pM called the tangent space of M at p. The tangent space is a flat plane consisting of all tangent

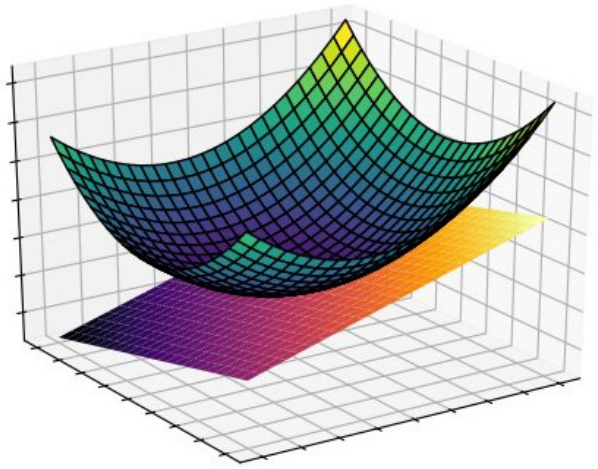


Figure 2.1: Example of the Tangent Space (orange shades) at a point on a 2-dimensional manifold (green shades) (best viewed in colour)

vectors to the manifold at the point p. The dimension of the tangent space at every point of a connected manifold is the same as that of the manifold itself [Carmo 1992]. An example of the tangent space at a point on a 2-dimensional manifold embedded in 3-dimensional space is shown in Figure 2.1. A Riemannian metric g assigns to each $p \in M$ a positive-definite inner product over the tangent space $g_p : T_pM \times T_pM \to \mathbb{R}$. The metric, thus, assigns a positive value to every non-zero tangent vector [Dodson and Poston 2013]. A real, smooth manifold M with a Riemannian metric g is a Riemannian manifold denoted as (M, g). The most familiar example of a Riemannian manifold is Euclidean space. Let $y^1, ..., y^n$ denote the Cartesian coordinates on $\mathbb{R}^n$. Then the metric tensor is $g_p = \mathbf{I} \in \mathbb{R}^{n \times n}_+$ and the inner-product is the familiar definition of the dot-product for Euclidean geometry $\sqrt{g_p(\mathbf{u}, \mathbf{v})} = \sqrt{\mathbf{u}^T \mathbf{I} \mathbf{v}} = \sqrt{\sum_i u_i v_i}$. For $\mathbb{R}^2$ this results in the traditional Pythagorean theorem. Thus, the metric generalizes the calculation of distance to non-Euclidean spaces and to arbitrary dimensions. The identity matrix is also called the canonical Euclidean metric.

This inner product can be used to define a norm for tangent vectors $|\cdot|_p : T_pM \to \mathbb{R}$ by $|\mathbf{v}|_p = \sqrt{g_p(\mathbf{v}, \mathbf{v})}$, as well as other geometric notions such as angles between vectors and the local volume of the manifold [Carmo 1992; Amari 2016]. The local volume of a manifold was used in Section 2.2 as the determinant of the metric tensor in the Jeffreys prior. For example we can look at the UV coordinate system where $UV \subseteq \mathbb{R}^2$. Let $\mathbf{a} = a_1 \frac{\partial}{\partial u} + a_2 \frac{\partial}{\partial v}$ and $\mathbf{b} = b_1 \frac{\partial}{\partial u} + b_2 \frac{\partial}{\partial v}$ where $\frac{\partial}{\partial u}, \frac{\partial}{\partial v}$ are the basis vectors of some tangent space and $a_1, a_2, b_1, b_2 \in \mathbb{R}$. Then, using the bilinearity of the dot product :

$$\mathbf{a} \cdot \mathbf{b} = a_1 b_1 \frac{\partial}{\partial u} \cdot \frac{\partial}{\partial u} + a_1 b_2 \frac{\partial}{\partial u} \cdot \frac{\partial}{\partial v} + a_2 b_1 \frac{\partial}{\partial u} \cdot \frac{\partial}{\partial v} + a_2 b_2 \frac{\partial}{\partial v} \cdot \frac{\partial}{\partial v} = \begin{bmatrix} a_1 & a_2 \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial u} \cdot \frac{\partial}{\partial u} & \frac{\partial}{\partial u} \cdot \frac{\partial}{\partial v} \\ \frac{\partial}{\partial u} \cdot \frac{\partial}{\partial v} & \frac{\partial}{\partial v} \cdot \frac{\partial}{\partial v} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}.$$

In this case the metric tensor is $g = \begin{bmatrix} \frac{\partial}{\partial u} \cdot \frac{\partial}{\partial u} & \frac{\partial}{\partial u} \cdot \frac{\partial}{\partial v} \\ \frac{\partial}{\partial u} \cdot \frac{\partial}{\partial v} & \frac{\partial}{\partial v} \cdot \frac{\partial}{\partial v} \end{bmatrix}$, $\mathbf{a} \cdot \mathbf{b} = g(\mathbf{a}, \mathbf{b})$, the norm of a tangent

vector is $|\mathbf{a}| = \sqrt{g(\mathbf{a},\mathbf{a})}$ and the angle θ between two tangent vectors is $cos(\theta) = \frac{g(\mathbf{a},\mathbf{b})}{|\mathbf{a}||\mathbf{b}|}$.

When $\begin{bmatrix} a_1 & a_2 \end{bmatrix} = \begin{bmatrix} b_1 & b_2 \end{bmatrix} = \begin{bmatrix} du & dv \end{bmatrix}$ then the inner-product describes the instantaneous change along the tangent space: $ds^2 = \begin{bmatrix} du & dv \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial u} \cdot \frac{\partial}{\partial u} & \frac{\partial}{\partial u} \cdot \frac{\partial}{\partial v} \\ \frac{\partial}{\partial u} \cdot \frac{\partial}{\partial v} & \frac{\partial}{\partial v} \cdot \frac{\partial}{\partial v} \end{bmatrix} \begin{bmatrix} du \\ dv \end{bmatrix}$. We also note that by the chain rule $\begin{bmatrix} du \\ dv \end{bmatrix} = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix} \begin{bmatrix} dx \\ dy \end{bmatrix}$ where dx and dy are changes in the XY coordinate system. Thus:

$ds^2 = \begin{bmatrix} du & dv \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial u} \cdot \frac{\partial}{\partial u} & \frac{\partial}{\partial u} \cdot \frac{\partial}{\partial v} \\ \frac{\partial}{\partial u} \cdot \frac{\partial}{\partial v} & \frac{\partial}{\partial v} \cdot \frac{\partial}{\partial v} \end{bmatrix} \begin{bmatrix} du \\ dv \end{bmatrix} = \begin{bmatrix} dx & dy \end{bmatrix} \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix}^T \begin{bmatrix} \frac{\partial}{\partial u} \cdot \frac{\partial}{\partial u} & \frac{\partial}{\partial u} \cdot \frac{\partial}{\partial v} \\ \frac{\partial}{\partial u} \cdot \frac{\partial}{\partial v} & \frac{\partial}{\partial v} \cdot \frac{\partial}{\partial v} \end{bmatrix} \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix} \begin{bmatrix} dx \\ dy \end{bmatrix}$

$= \begin{bmatrix} dx & dy \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial x} \cdot \frac{\partial}{\partial x} & \frac{\partial}{\partial x} \cdot \frac{\partial}{\partial y} \\ \frac{\partial}{\partial x} \cdot \frac{\partial}{\partial y} & \frac{\partial}{\partial y} \cdot \frac{\partial}{\partial y} \end{bmatrix} \begin{bmatrix} dx \\ dy \end{bmatrix} = (ds')^2$. This reflects two important points. Firstly that the distance calculation with the metric tensor is invariant under a change of coordinate systems. Secondly that the metric tensor transforms using the Jacobian matrix $\mathbf{J_f}$, the matrix of all first-order partial derivatives of a function $\mathbf{f} : XY \to UV$ that maps between coordinate spaces. Thus $g' = {\mathbf{J_f}}^T g \mathbf{J_f}$ where g' is the metric tensor in the XY coordinate system and is known as the pullback metric of g (for UV space) from $\mathbf{f}$. The invariance of the metric tensor to a change of coordinates allows us to compute the inner product of tangent vectors in a manner independent of the parametric description of the manifold and so we can define a notion of distance in the UV space but calculate the distance in the XY coordinate system.

We have described the relationship between a manifold, its metric tensor and tangent bundle (set of all tangent spaces on the manifold), as well as how to map between different coordinate spaces. The final step is to describe how to obtain a metric tensor to begin with. Regions of a manifold are described using charts, where one or more charts are combined to form an atlas [Jost 2008; Gauld 1974]. A chart for a Riemannian manifold (M, g) is a diffeomorphism φ from an open subset $U \subset M$ to an open subset of a Euclidean space. The chart is denoted as the ordered pair (U, φ). Since charts map onto Euclidean space we can then use the canonical Euclidean metric and also transform the points on the manifold to points in this Euclidean space where we can calculate distance. However if we wish to still work in the original space with the defined notion of distance in Euclidean space, we may then pullback the metric from the Euclidean space using the locally defined chart to obtain a metric tensor for the tangent space of the manifold. It is important to note that the pullback metric is not necessarily positive definite and so the manifold described by the pullback metric may not be Riemannian, but rather pseudo-Riemannian [Jost 2008]. Pseudo-Riemannian manifolds are a generalization of Riemannian manifolds where the positive definite constraint on the metric tensor is relaxed such that the metric tensor is only required to be everywhere non-degenerate. The non-degenerate property means that there is no non-zero $\mathbf{X} \in T_pM$ such that $g_p(\mathbf{X}, \mathbf{Y}) = 0$ for all $\mathbf{Y} \in T_pM$. This is in contrast to the positive-definite constraint where for a non-zero $\mathbf{X} \in T_pM$ then $g(\mathbf{X}, \mathbf{Y}) \neq 0$ for all $\mathbf{Y} \in T_pM$.

Specifically for the case of NNs, we note that the parameter space of a probabilistic model forms a statistical manifold and by extension a Riemannian manifold [Rao 1945]. The metric tensor for statistical manifolds is the Fisher Information Matrix [Skovgaard 1984; Amari 2016] shown earlier in Equation 2.3. If we interpret the loss function as a coordinate chart from the N-dimensional parameter space onto the 1-dimensional Euclidean space then the Fisher Information Matrix is viewed as the pullback metric of the canonical Euclidean metric from the expected value of the loss. To show this we make use of the fact that since the loss is 1-dimensional then $L = \theta^T J_L$ which is the usual notion of the differential of L. Hence: $E_{\mathbf{X}}[L^2] = E_{\mathbf{X}}[\theta^T J_L J_L^T \theta] = \theta^T E_{\mathbf{X}}[J_L J_L^T]\theta = \theta^T I(\theta)\theta$

The equality of the Fisher Information Matrix and Hessian matrix is also significant as the Hessian is used in the area of a critical point on a Riemannian manifold to obtain the principal curvatures [Porteous 2001], and as a result the shape operator at that point [Spivak 1970]. In the case of a Riemannian manifold the shape operator is defined as the determinant of the Hessian matrix $det(H(\theta))$ [Koenderink and Van Doorn 1992]. The principal curvatures are defined as the eigenvectors of the Hessian matrix and decompose the manifold into orthogonal dimensions of curvature, with the first eigenvector reflecting the dimension of most curvature. The eigenvalues from the eigen-decomposition of the Hessian is known as the spectrum of the Hessian and the determinant of the Hessian can be calculated from the product of its eigenvalues. The Fisher Information Matrix (and by extension Hessian) is also significant as it upper-bounds the inverse covariance matrix of the parameters of a statistical model through the Cramér-Rao Lower bound which states that $\Sigma^{-1} \leqslant I(\theta)$. As a result larger elements of the Fisher Information Matrix reflect that there is a lower degree of variance in the model parameters and more certainty that those parameters are necessary for fitting the training data. As stated in Chapter 1 we believe this to be useful information for creating a regularizer by using the Riemannian distance to restrict the parameters of an NN. In light of this the Metric regularizer can be viewed as regularizing the low variance parameters of an NN more strictly, while the more irrelevant and high variance parameters are regularized less.

The Hessian matrix also provides the link between Riemannian Geometry and previous work with NNs as there has been a significant amount of work exploring the Hessian of the loss function in recent years. Most of these works have centered around the eigen-decomposition of the Hessian and the impact of flat regions of the landscape on both first and second-order training methods. Examples include the observation that the spectrum of the Hessian contains a few, large Eigenvalues but the large majority of Eigenvalues are near zero [Sagun *et al.* 2017]. It was also observed that various changes to the hyper-parameters during training change the final parametrization learned by the model, but that these minima tended to lie along the same connected basin in the loss landscape. This was extended by Gur-Ari *et al.* [2018] which empirically studied the overlap of the direction followed by GD and the

directions corresponding to the large (top) Eigenvalues of the Hessian. It was found that after a brief warm-up period GD tended to follow these top directions nearly exclusively. There has also been work that aims to utilize the second-order information to create new practical algorithms that are more suited to optimizing non-convex manifolds. In Dauphin *et al.* [2014] the gradients of normal GD update steps are multiplied by the inverse of the absolute values of the Hessian matrix as well as the top Lanczos vectors of the Hessian. This creates an update step that is well adapted to escape saddle points within the loss landscape but is still able to effectively optimize the manifold.

A common consideration of all previous work regarding the Hessian is that the loss landscape contains flat regions along which the parameters of the network can be changed without affecting the network's loss. This results in the Hessian matrix being singular. These flat regions are a result of covariant parameters that may be changed in a coordinated manner without changing the function approximation of the network. Such covariant parameters naturally occur with the addition of hidden layers to the model, resulting in parameters along the same path through the network becoming correlated. In addition the increase in width of the layers results in neurons of the same layer learning similar mappings and becoming interchangeable. While flat regions of the loss landscape reflect that the training loss is not affected by a change in parameter values along the flat path, different parametrizations may have significantly different performance on unseen data. This is because many network parametrizations can perfectly fit the seen data (data sampled from the data subspace) but perform different function approximations for unseen data (over the data null-space). The effective handling of the data null space is the point of regularization as it constrains the function approximation for unseen data. By creating a bias for the model parameters towards 0 we constrain the model to simpler function approximations that interpolate smoothly between the seen data points. The necessity of regularization for data-constrained problems can be seen from Bayes' rule where a prior is needed for data-constrained Bayesian inference. As the quantity of data increases ($P \to \infty$) the MAP solution converges to the ML solution, reflecting the decreasing need for the prior/regularization.

Chapter 3

Theoretical Analysis

3.1 Introduction

In this chapter we present the theoretical analysis of the proposed Metric regularization method, defined by the learning rule: $\theta_{(t+1)} \leftarrow \theta_{(t)} - \nabla_\theta L(\mathbf{X}, \theta_{(t)}) - \theta_{(t)}^T I(\theta_{(t)})$. The third term on the right side, $\theta_{(t)}^T I(\theta_{(t)})$, is the regularization part of the learning rule. This term is the first derivative of the Riemannian distance of the model parameters: $\nabla_\theta \theta_{(t)}^T I(\theta_{(t)}) \theta_{(t)}$. As shown in Section 2.4, the Riemannian distance allows us to define distance in the loss-space while being able to work in parameter space. By taking the derivative of the Riemannian distance we are able to find which parameters have the most impact on the loss of the model. Two primary benefits of using the Riemannian distance is that it incorporates the loss and activation functions used through the chain-rule, and so it can be used with any such functions as long as they are differentiable (a condition already required by gradient descent). Secondly, the impact of a parameter is determined by its interaction with the other parameters of the model. This is due to the dot-product of the parameters and Fisher Information Matrix: $\theta_{(t)}^T I(\theta_{(t)})$. By accounting for the interactions between parameters we aim to generalize regularization to correlated parameter spaces.

In particular there are two related effects being observed in the Fisher Information Matrix. The diagonal reflects the sensitivity of a parameter in isolation, while the off-diagonal elements reflect the sensitivity of a parameters due to its interaction with another parameter. Naturally the sensitivity of a parameter in isolation will be related to its sensitivity due to interactions. Parameters which are sensitive in isolation are likely to be sensitive when interacting with other parameters, however, it is possible for the diagonal element for a parameter to be insensitive (small) while its off-diagonal elements are sensitive (large). This will be most likely when parameters have very different scales. Hence, it is particularly important to account for the correlation between parameters when we the parameters are large. As we will see in Chapter 5, this intuition fore-shadows our empirical results.

We begin in Section 3.2 where we show that the parameters of separate layers in an NN are correlated, motivating the use of regularizers that account for this fact. In Section 3.3 we then derive a useful bound showing the effect using the Cramér-Rao Lower Bound has on the shape of a Multivariate Gaussian distribution. Section 3.3 also highlights the relationship between the Fisher Information Matrix and the covariance matrix of the Multivariate Gaussian distribution and shows how the Fisher Information Matrix can reflect the correlation between parameters. Section 3.4 then follows with the derivations of the Fisher Information Matrix for the relevant probability distributions that are used in the following sections. In Section 3.5 we then derive the regularizer from a Bayesian prior in the same manner as the derivations of L1 and L2 regularization in Section 2.3. Section 3.6 then reflects that the Bayesian parameter approximation from the Multivariate Gaussian prior with the Gaussian likelihood distribution is equivalent to the Minimum Mean Squared Estimation (MMSE) for the parameters of a linear regression model. The MMSE parameters for linear regression result from the MAP inference with an isotropic (which implies independence) Gaussian prior [Advani *et al.* 2013; Advani and Ganguli 2016]. We show an equivalence between using the Multivariate Gaussian with an NN and an independent Gaussian with linear regression. Section 3.7 then reflects that the MAP parameter inference from Section 3.6 is equivalent to the Bayesian inference in Section 3.5. Sections 3.5, 3.6 and 3.7, thus, reflect that the stable point parametrization from the Metric regularizer has the MMSE property. The final two sections, Sections 3.8 and 3.9 discuss the feasibility of performing Metric regularization as well as calculating the Eigen-spectrum of the Hessian matrix of large NNs. In these sections we derive a means to do both and present a novel algorithm for computing the Eigen-spectrum of the Hessian of an NN. The proof that Metric regularization results in the MMSE parametrization in Sections 3.5, 3.6 and 3.7, as well as the theory to make Metric regularization and finding the Eigen-spectrum feasible in Sections 3.8 and 3.9 are the two primary contributions of this chapter. The primary theorem of this chapter is stated as:

Theorem 1: Assuming a twice-differentiable loss function Metric regularization provides the Minimum Mean Squared Error parameters for a Neural Network.

3.2 Parameters Along a Path are Correlated

In this section, we show that all parameters which are connected by at least one path through the network are correlated. This motivates the need to account for the correlation between parameters in deep networks. To show this we use a proof by induction where we show that regardless of the length of a path through the network, a parameter is always in the derivative of the other parameters along the same path. In fully-connected networks this means that all parameters which are not a part of the same layer are correlated. It is possible for parameters of the same layer to be correlated but not guaranteed, and so we ignore it in this section.

We do not make claims in this section about the strength of the correlations or how the

strength are affected by increasing the network width or depth. Indeed with complex enough input data and enough parameter interactions in a large network it is unlikely that any two parameters will be perfectly correlated. Nonetheless, the aim of this section is to theoretically motivate the removal of independence assumptions in regularizers, which is not completely justified in the presence of any parameter correlation. We begin again by using the quadratic loss function:

Let $L = \frac{1}{2}(y - f(\theta, x))^2$

Let $f(\theta, x) = \phi(\theta_2\phi(\theta_1 x))$ where ϕ is any activation function.

Thus, $f(\theta, x)$ is a single pathway network with one hidden layer.

Then $\nabla_{\theta_1} L = \nabla_{\theta_1} \frac{1}{2}(y - \phi(\theta_2\phi(\theta_1 x)))^2$

$$= -(y - \phi(\theta_2\phi(\theta_1 x)))\phi'(\theta_2\phi(\theta_1 x))\phi'(\theta_1 x)x$$
$$= (\phi(\theta_2\phi(\theta_1 x)) - y)\phi'(\theta_2\phi(\theta_1 x))\phi'(\theta_1 x)x$$
$$= \phi(\theta_2\phi(\theta_1 x))\phi'(\theta_2\phi(\theta_1 x))\phi'(\theta_1 x)x - y\phi'(\theta_2\phi(\theta_1 x))\phi'(\theta_1 x)x$$

A minimum occurs when $\nabla_{\theta_1} L = 0$

$$\phi(\theta_2\phi(\theta_1 x))\phi'(\theta_2\phi(\theta_1 x))\phi'(\theta_1 x)x - y\phi'(\theta_2\phi(\theta_1 x))\phi'(\theta_1 x)x = 0$$
$$\phi(\theta_2\phi(\theta_1 x))\phi'(\theta_2\phi(\theta_1 x))\phi'(\theta_1 x)x = y\phi'(\theta_2\phi(\theta_1 x))\phi'(\theta_1 x)x$$
$$\phi(\theta_2\phi(\theta_1 x)) = y$$
$$\phi(\theta_1 x) = \frac{\phi^{-1}(y)}{\theta_2}$$
$$\theta_1 = \phi^{-1}\left(\frac{\phi^{-1}(y)}{\theta_2}\right) x^{-1}$$

Therefore the parameters of a one-hidden layer path are correlated and the base case holds.

Inductive Hypothesis: Assume that θ_1 is correlated to the parameters of a single path network of depth N

Inductive Step: We then show that θ_1 is correlated to the parameters when depth is $N + 1$

Let $\bar{\phi}$ represent the $N - 1$ parameters and activations between the first and last parameters.

Then $\nabla_{\theta_1} L = \nabla_{\theta_1} \frac{1}{2}(y - \phi(\theta_{N+1}\bar{\phi}(\theta_1 x)))^2$

A minimum occurs when $\nabla_{\theta_1} L = 0$

$$\phi(\theta_{N+1}\bar{\phi}(\theta_1 x))\phi'(\theta_{N+1}\bar{\phi}(\theta_1 x))\bar{\phi}'(\theta_1 x)x - y\phi'(\theta_{N+1}\bar{\phi}(\theta_1 x))\bar{\phi}'(\theta_1 x)x = 0$$
$$\phi(\theta_{N+1}\bar{\phi}(\theta_1 x)) = y$$
$$\phi(\theta_1 x) = \frac{\bar{\phi}^{-1}(y)}{\theta_{N+1}}$$

$$\theta_1 = \phi^{-1}\left(\frac{\bar{\phi}^{-1}(y)}{\theta_{N+1}}\right)x^{-1}$$

Therefore, the parameters along a path through the network are correlated since the final value of θ_1 depends on the N other parameters along the path. Proved by induction.

3.3 Effect of Cramér-Rao Lower Bound on Gaussian Distribution

In this section we prove an inequality that results by using the Cramér-Rao Lower Bound on a Multivariate Gaussian distribution. In particular we prove a condition on θ for where, in parameter space, a Multivariate Gaussian distribution will place more density than if its covariance were replaced by the inverse Fisher Information Matrix. This shows that the inverse Fisher Information Matrix compresses the density of the Multivariate Gaussian closer around the mean and that the true Multivariate Gaussian (with the covariance matrix) has longer tails to its distribution. We derive the inequality in this section so that we are aware of which regions of parameter space are most affected by the change from the inverse-covariance matrix to Fisher Information Matrix. In particular we derive an inequality which shows that as the parameter values θ get larger the distribution with the Fisher Information Matrix lower-bounds the distribution with the inverse-covariance matrix. This is a desirable property since we use the Fisher Information distribution to derive Metric regularization. This shows that the regularizer will be most effective in the regions where its is most necessary, where the parameter values are large.

Consider the case when:

$$\begin{aligned}
\frac{1}{\sqrt{(2\pi)^N|\Sigma|}}\exp\left(-\frac{1}{2}\theta^T\Sigma^{-1}\theta\right) &\geqslant \frac{1}{\sqrt{(2\pi)^N}}\exp\left(-\frac{1}{2}\theta^T I(\theta)\theta\right)\sqrt{|I(\theta)|} \\
\exp\left(-\frac{1}{2}\theta^T\Sigma^{-1}\theta\right) &\geqslant \exp\left(-\frac{1}{2}\theta^T I(\theta)\theta\right)\sqrt{|I(\theta)||\Sigma|} \\
\exp\left(-\frac{1}{2}\theta^T\Sigma^{-1}\theta + \frac{1}{2}\theta^T I(\theta)\theta\right) &\geqslant \sqrt{|I(\theta)\Sigma|} \\
\exp\left(\frac{1}{2}\theta^T\left(I(\theta) - \Sigma^{-1}\right)\theta\right) &\geqslant \sqrt{|I(\theta)\Sigma|} \\
\frac{1}{2}\theta^T\left(I(\theta) - \Sigma^{-1}\right)\theta &\geqslant \frac{1}{2}\log\left(|I(\theta)\Sigma|\right) \\
\frac{\theta^T\left(I(\theta) - \Sigma^{-1}\right)\theta}{\log\left(|I(\theta)\Sigma|\right)} &\geqslant 1
\end{aligned}$$

From the Cramér-Rao lower bound we know that $\Sigma \geqslant I(\theta)^{-1} \Leftrightarrow \Sigma^{-1} \leqslant I(\theta)$

As a result: $I(\theta) - \Sigma^{-1} \geqslant I(\theta) - I(\theta)^{-1} = \mathbf{0}$

Similarly: $I(\theta)\Sigma \geqslant I(\theta)I(\theta)^{-1} = \mathbf{I}$

So $\log(|I(\theta)\Sigma|)$ is defined and $\log(|I(\theta)\Sigma|) \geqslant \log(|\mathbf{I}|) \geqslant 0$

Thus, all terms in the expression are positive and well-defined, with the inequality depending on θ.

3.4 Fisher Information of Gaussian Likelihood and Posterior Distributions

In this section we present the calculations of the Fisher Information Matrices for three distributions. The Multivariate Gaussian likelihood distribution in Equation 3.1, the joint likelihood distribution of the data and hyper-parameters conditioned on the model parameters in Equation 3.2 and lastly the posterior distribution over the model parameters conditioned on the data and hyper-parameters in Equations 3.3. We use all three versions of the Fisher Information Matrix in Sections 3.5, 3.6 and 3.7.

Let:

$$P(\mathbf{X}|\theta) = \frac{1}{Z}\exp\left(-\sum_{i=1}^{P}\frac{(y_i - f(x_i,\theta))^2}{2\sigma_i^2}\right) \tag{3.1}$$

Then the Fisher Information Matrix of $P(\mathbf{X}|\theta)$ can be calculated as follows:

$I^*(\theta) = -\nabla_\theta\nabla_\theta \log(P(\mathbf{X}|\theta))$

We start by applying $-\log$ to $P(\mathbf{X}|\theta)$:

$$-\log(P(\mathbf{X}|\theta)) = \sum_{i=1}^{P}\frac{(y_i - f(x_i,\theta))^2}{2\sigma_i^2} + \log(Z)$$

$$I^*(\theta) = \frac{1}{2\sigma_i^2}\sum_{i=1}^{P}\nabla_\theta\nabla_\theta(y_i - f(x_i,\theta))^2 + \nabla_\theta\nabla_\theta\log(Z) = \frac{1}{2\sigma_i^2}\sum_{i=1}^{P}\nabla_\theta\nabla_\theta(y_i - f(x_i,\theta))^2$$

$$I^*(\theta) = -\frac{1}{\sigma^2}\sum_{i=1}^{P}\nabla_\theta\left((y_i - f(x_i,\theta))\nabla_\theta f(x_i,\theta)\right) = -\frac{1}{\sigma^2}\sum_{i=1}^{P}\nabla_\theta\left(y_i\nabla_\theta f(x_i,\theta) - f(x_i,\theta)\nabla_\theta f(x_i,\theta)\right)$$

$$I^*(\theta) = -\frac{1}{\sigma^2}\sum_{i=1}^{P}\left(y_i\nabla_\theta\nabla_\theta f(x_i,\theta) - \nabla_\theta f(x_i,\theta)^2 - f(x_i,\theta)\nabla_\theta\nabla_\theta f(x_i,\theta)\right)$$

We can also include the hyper-parameter distribution, where θ^* is the hyper-parameter that we set to $\mathbf{0}$ for the remainder of this work.

$$P(\theta^*|\theta) = \frac{1}{\sqrt{(2\pi)^N|\Sigma|}}\exp\left(-\frac{1}{2}(\theta^* - \theta)^T\Sigma^{-1}(\theta^* - \theta)\right)$$

Using the Cramér-Rao lower bound, which states that $\Sigma \geqslant I(\theta)^{-1} \Leftrightarrow \Sigma^{-1} \leqslant I(\theta)$ with the assumption that $\frac{\theta^T\left(I^*(\theta) - \Sigma^{-1}\right)\theta}{\log|I^*(\theta)\Sigma|} \geqslant 1$ (see Section 3.3 for the derivation of this inequality and its utility) we have the following:

$\frac{1}{\sqrt{(2\pi)^N|\Sigma|}}\exp\left(-\frac{1}{2}\theta^T\Sigma^{-1}\theta\right) \geqslant \frac{1}{\sqrt{(2\pi)^N}}\exp\left(-\frac{1}{2}\theta^T I^*(\theta)\theta\right)\sqrt{|I^*(\theta)|}$ such that

$$P(\mathbf{X}, \theta^*|\theta) = \frac{1}{Z(2\pi)^{\frac{N}{2}}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{P}(y_i - f(x_i,\theta))^2 - \frac{1}{2}\theta^T I^*(\theta)\theta + \frac{1}{2}\log(|I^*(\theta)|)\right) \tag{3.2}$$

Then the Fisher Information Matrix of $P(\mathbf{X}, \theta^*|\theta)$ can be calculated as follows:

$I(\theta) = -\nabla_\theta \nabla_\theta \log(P(\mathbf{X}, \theta|\theta))$ where the first step is to take the $-\log$ of Equation 3.2:

$$-\log(P(\mathbf{X}, \theta^*|\theta)) = \frac{1}{2\sigma^2}\sum_{i=1}^{P}(y_i - f(x_i,\theta))^2 + \frac{1}{2}\theta^T I^*(\theta)\theta - \frac{1}{2}\log(|I^*(\theta)|) + \log(Z(2\pi)$$

$$I(\theta) = \frac{1}{2\sigma^2}\sum_{i=1}^{P}\nabla_\theta\nabla_\theta(y_i - f(x_i,\theta))^2 + \frac{1}{2}\nabla_\theta\nabla_\theta\theta^T I^*(\theta)\theta - \frac{1}{2}\nabla_\theta\nabla_\theta\log(|I^*(\theta)|) + \nabla_\theta\nabla_\theta\log(Z(2\pi)$$

$$I(\theta) = I^*(\theta) + \frac{1}{2}I^*(\theta) - \frac{1}{2}\nabla_\theta\nabla_\theta\log(|I^*(\theta)|)$$

Assuming that the second derivatives are constant means that $\nabla_\theta\nabla_\theta\log(|I^*(\theta)|) = 0$

$$I(\theta) = \frac{3}{2}I^*(\theta)$$

Finally, we can include the prior distribution:

$P(\theta) = \sqrt{|I(\theta)|}$ such that the posterior distribution is:

$$P(\theta|\mathbf{X}, \theta^*) = \frac{1}{Z(2\pi)^{\frac{N}{2}}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{P}(y_i - f(x_i,\theta))^2 - \frac{1}{2}\theta^T I^*(\theta)\theta\right)\left(\sqrt{|I^*(\theta)|}\right)\left(\sqrt{|I(\theta)|}\right)$$

$$\int P(\theta|\mathbf{X}, \theta^*)d\theta = \frac{1}{Z(2\pi)^{\frac{N}{2}}}\int \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{P}(y_i - f(x_i,\theta))^2 - \frac{1}{2}\theta^T I^*(\theta)\theta + \frac{1}{2}\log(|I^*(\theta)|) + \frac{1}{2}\log(|I(\theta)|)\right)d\theta \tag{3.3}$$

Let $L(\theta) = \frac{1}{2\sigma^2}\sum_{i=1}^{P}(y_i - f(x_i,\theta))^2 + \frac{1}{2}\theta^T I^*(\theta)\theta - \frac{1}{2}\log(|I^*(\theta)|) - \frac{1}{2}\log(|I(\theta)|)$ then

$$\nabla_\theta\nabla_\theta L(\theta) = \frac{1}{2\sigma^2}\sum_{i=1}^{P}\nabla_\theta\nabla_\theta(y_i - f(x_i,\theta))^2 + \frac{1}{2}\nabla_\theta\nabla_\theta\theta^T I^*(\theta)\theta - \frac{1}{2}\nabla_\theta\nabla_\theta\log(|I^*(\theta)|) - \frac{1}{2}\nabla_\theta\nabla_\theta\log(|I(\theta)|)$$

$$\nabla_\theta\nabla_\theta L(\theta) = I(\theta) - \frac{1}{2}\nabla_\theta\nabla_\theta\log(|I(\theta)|)$$

Again assuming that the second derivatives are constant, we have:

$\nabla_\theta\nabla_\theta L(\theta) = I(\theta)$

For this work we are primarily working with the integral: $\int P(\theta|\mathbf{X}, \theta^*)d\theta$

and its use in deriving a new regularization method for artificial neural networks.

3.5 Derivation of Metric Regularizer from Bayesian Prior

Now that we have motivated the use of regularization methods that account for the dependency between parameters and calculated the necessary Fisher Information Matrices we can begin to show that the Metric regularization method finds the Minimum Mean Squared

Error (MMSE) parametrization of an NN. We begin by deriving Metric regularization from a Bayesian prior, similar to the derivations of L1 and L2 regularization in Section 2.3. Thus, by minimizing the error in Equation 3.5 we are finding the MAP solution for the parameters. The primary lemma of this section is:

Lemma 1: Assuming constant second derivatives Metric regularization provides the Maximum A Posteriori solution for a Multivariate Gaussian Prior in a Bayesian framework.

We use both the Gaussian likelihood and hyper-parameter distributions for this derivation.

We begin with:

$$P(\mathbf{X}, \theta^*|\theta) = \frac{1}{Z(2\pi)^{\frac{N}{2}}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{P}(y_i - f(x_i, \theta))^2 - \frac{1}{2}\theta^T I^*(\theta)\theta + \frac{1}{2}\log(|I^*(\theta)|)\right) \tag{3.4}$$

Taking the negative of the natural log of this joint likelihood distribution we obtain:

$$L(\theta) = -\log P(\mathbf{X}, \theta^*|\theta) = \frac{1}{2\sigma^2}\sum_{i=1}^{P}(y_i - f(x_i, \theta))^2 + \frac{1}{2}\theta^T I^*(\theta)\theta - \frac{1}{2}\log(|I^*(\theta)|) + \log(Z(2\pi)^{\frac{N}{2}})$$

Since $\log(Z(2\pi)^{\frac{N}{2}})$ is independent of the θ value it does not change the stable point of θ and so we can drop it from the loss function.

This gives us the loss function:

$$L(\theta) = \frac{1}{2\sigma^2}\sum_{i=1}^{P}(y_i - f(x_i, \theta))^2 + \frac{1}{2}\theta^T I^*(\theta)\theta - \frac{1}{2}\log(|I^*(\theta)|) \tag{3.5}$$

We aim to minimize this function with regard to the parameters.

We, thus, take the derivative with respect to θ to find its stable points.

$$\nabla_\theta L(\theta) = \frac{1}{2\sigma^2}\sum_{i=1}^{P}\nabla_\theta(y_i - f(x_i, \theta))^2 + \frac{1}{2}\nabla_\theta\theta^T I^*(\theta)\theta - \frac{1}{2}\nabla_\theta\log(|I^*(\theta)|)$$

$$\nabla_\theta L(\theta) = -\frac{1}{\sigma_i^2}\sum_{i=1}^{P}(y_i - f(x_i, \theta))(\nabla_\theta f(x_i, \theta)) + \frac{1}{2}\theta^T I^*(\theta) - \frac{1}{2}Tr([I^*(\theta)^{-1}]T^*(\theta))$$

where $T^*(\theta)$ is the tensor of third derivatives and the Trace is over individual 2-D slices of the 3-D output tensor.

In practice a constant is applied to the regularizer to determine the degree of regularization.

We denote this as λ. In addition we now include the learning rate α.

We then have the parameter update equation:

$$\theta_{i(t+1)} = \theta_{i(t)} + \alpha'\sum_{i=1}^{P}(y_i - f(x_i, \theta))(\nabla_\theta f(x_i, \theta)) - \lambda'\theta^T I^*(\theta) + \lambda' Tr([I^*(\theta)^{-1}]T^*(\theta))$$

Where $\alpha' = \frac{\alpha}{\sigma_i^2}$ and $\lambda' = \frac{\lambda}{2}$ since σ_i^2 is assumed constants with respect to θ.

Assuming again that the second derivatives are constant we have:

$$\theta_{i(t+1)} = \theta_{i(t)} + \alpha' \sum_{i=1}^{P} (y_i - f(x_i, \theta))(\nabla_\theta f(x_i, \theta)) - \lambda' \theta^T I^*(\theta))$$

This is the learning rule shown in at the start of the chapter with the quadratic loss being used as a consequence of the Gaussian likelihood distribution being used initially.

3.6 Equivalence to MMSE for Linear Regression

In Section 3.5 we derived Metric regularization as the learning rule which maximizes the likelihood of the joint distribution of the data and hyper-parameters dependent on the model parameters: $P(\mathbf{X}, \theta^*|\theta)$. This is equal to the traditional MAP parametrization. The necessity to distinguish between a hyper-parameter distribution and then use a Jeffreys prior, as we have set up in Section 3.4, will be made clear in Section 3.7. In this section we include the Jeffreys prior and work with the posterior distribution $P(\theta|\mathbf{X}, \theta^*)$. We show that using this posterior distribution over the parameter space of an NN is equivalent to using an independent/isotropic Multivariate Gaussian prior to derive the Bayesian parametrization of a linear regression model. This has been shown to provide the MMSE parameters for a linear regression model [Advani and Ganguli 2016]. Thus, $P(\theta|\mathbf{X}, \theta^*)$ will provide the MMSE parametrization for an NN, under the assumption of a Gaussian likelihood and hyper-parameter distribution and the implied assumption that the parameter covariance is constant. The primary lemma of this section is:

Lemma 2: Assuming a Gaussian likelihood and hyper-parameter distribution, and constant second derivatives the Bayesian posterior distribution with a Jeffreys prior is equivalent to the MMSE linear-regression model.

We begin using the Gaussian likelihood and hyper-parameter distributions with a Jeffreys prior as described above. As a result we have the posterior distribution:

$$P(\theta|\mathbf{X}, \theta^*) = P(\mathbf{X}|\theta)P(\theta^*|\theta)P(\theta)$$

$$P(\theta|\mathbf{X}, \theta^*) = \frac{1}{Z(2\pi)^{\frac{N}{2}}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{P}(y_i - f(x_i, \theta))^2 - \frac{1}{2}(\theta^* - \theta)^T I^*(\theta)(\theta^* - \theta)\right)\left(\sqrt{|I^*(\theta)|}\right)\left(\sqrt{|I(\theta)|}\right)$$

Thus:

$$\int P(\theta|\mathbf{X}, \theta^*)d\theta = \int \frac{1}{Z(2\pi)^{\frac{N}{2}}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{P}(y_i - f(x_i, \theta))^2 - \frac{1}{2}(\theta^* - \theta)^T I^*(\theta)(\theta^* - \theta)\right)\left(\sqrt{|I^*(\theta)|}\right)\left(\sqrt{|I(\theta)|}\right)d\theta$$

$$\int P(\theta|\mathbf{X}, \theta^*)d\theta = \int \frac{1}{Z(2\pi)^{\frac{N}{2}}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{P}(y_i - f(x_i, \theta))^2 - \frac{1}{2}(\theta^* - \theta)^T I^*(\theta)(\theta^* - \theta)\right)\left(\sqrt{|I^*(\theta)|}\right)\left(\sqrt{\frac{3}{2}^N |I^*(\theta)|}\right)d\theta$$

Using the Eigen-decomposition we get: $I^*(\theta) = Q\Lambda Q^{-1}$ where Q is the ortho-normal matrix of Eigenvectors and

Λ is the diagonal matrix of Eigenvalues. Thus $\Lambda = Q^{-1} I^*(\theta) Q$.

We now use the change of variables formula to project the network parameters onto the ortho-normal basis Q.

The change of variables formula is as follows:

$$\int P(\theta|\mathbf{X}, \theta^*) d\theta = \int P(\mu|\mathbf{X}, \mu^*) |J_\mu(\theta)| d\mu$$

Where μ is the new parameter set after the projection, and $J_\mu(\theta)$ is the Jacobian matrix of θ with respect to μ.

We have: $\mu = Q^{-1}\theta$ and $\mu^* = Q^{-1}\theta^*$. Similarly, $\theta = Q\mu$. Thus, $J_\mu(\theta) = \nabla_\mu \theta = \nabla_\mu Q\mu = Q$

We also set $\theta^* = \mathbf{0}$ and so $\mu^* = Q\mathbf{0} = \mathbf{0}$. Due to the assumption of constant second derivatives, Q is also constant.

Using the change of variables formula with the Eigen-decomposition we obtain the following:

$$\begin{aligned}
\int P(\theta|\mathbf{X}, \theta^*) d\theta &= \int \frac{1}{Z(4/3\pi)^{\frac{N}{2}}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{P}(y_i - f(x_i, \theta))^2 - \frac{1}{2}(\theta)^T I^*(\theta)(\theta)\right)\left(\sqrt{|I^*(\theta)|}\right)\left(\sqrt{|I^*(\theta)|}\right)|J_\mu(\theta)| d\mu \\
&= \int \frac{1}{Z(4/3\pi)^{\frac{N}{2}}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{P}(y_i - f(x_i, Q\mu))^2 - \frac{1}{2}(Q\mu)^T I^*(\theta)(Q\mu)\right)\left(\sqrt{|I^*(\theta)|}\right)\left(\sqrt{|I^*(\theta)|}\right)|Q| d\mu \\
&= \int \frac{1}{Z(4/3\pi)^{\frac{N}{2}}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{P}(y_i - f(x_i, Q\mu))^2 - \frac{1}{2}\mu^T (Q^T I^*(\theta) Q)\mu\right)\left(\sqrt{|I^*(\theta)|}\right)\left(\sqrt{|I^*(\theta)||Q|^2}\right) d\mu \\
&= \int \frac{1}{Z(4/3\pi)^{\frac{N}{2}}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{P}(y_i - f(x_i, Q\mu))^2 - \frac{1}{2}\mu^T (Q^T I^*(\theta) Q)\mu\right)\left(\sqrt{|I^*(\theta)|}\right)\left(\sqrt{|Q^T I^*(\theta) Q|}\right) d\mu
\end{aligned}$$

We now use the fact that Q is an ortho-normal matrix, such that $Q^T = Q^{-1}$.

$$\begin{aligned}
&= \int \frac{1}{Z(4/3\pi)^{\frac{N}{2}}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{P}(y_i - f(x_i, Q\mu))^2 - \frac{1}{2}\mu^T (Q^{-1} I^*(\theta) Q)\mu\right)\left(\sqrt{|I^*(\theta)|}\right)\left(\sqrt{|Q^{-1} I^*(\theta) Q|}\right) d\mu \\
&= \int \frac{1}{Z(4/3\pi)^{\frac{N}{2}}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{P}(y_i - f(x_i, Q\mu))^2 - \frac{1}{2}\mu^T \Lambda \mu\right)\left(\sqrt{|I^*(\theta)|}\right)\left(\sqrt{|\Lambda|}\right) d\mu
\end{aligned}$$

Finally, let $|\Lambda| = C$ and $\Lambda = vIv^{-1}$ such that $I(\theta) = Q\Lambda Q^{-1} = \hat{Q} v I v^{-1} \hat{Q}^{-1}$. This is the rescaling of the Eigen-basis to result in unit Eigenvalues. We also note that $\sqrt{|I^*(\theta)|} = \sqrt{|Q\Lambda Q^{-1}|} = \sqrt{|Q||\Lambda||Q^T|} = \sqrt{|\Lambda|}$.

$$\begin{aligned}
&= \int \frac{1}{Z(4/3\pi)^{\frac{N}{2}}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{P}(y_i - f(x_i, Q\mu))^2 - \frac{1}{2}\mu^T v^T I v \mu\right)\left(\sqrt{|\Lambda|}\right)\left(\sqrt{C}\right) d\mu \\
&= \int \frac{1}{Z(4/3\pi)^{\frac{N}{2}}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{P}(y_i - f(x_i, Q\mu))^2 - \frac{1}{2}(v\mu)^T I v \mu\right)\left(\sqrt{C}\right)\left(\sqrt{C}\right) d\mu \\
&= \int \frac{C}{Z(4/3\pi)^{\frac{N}{2}}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{P}(y_i - f(x_i, Q\mu))^2 - \frac{1}{2}||v\mu||_2^2\right) d\mu
\end{aligned}$$

This equivalence relies on the Fisher Information Matrix being non-singular to compute the determinant.

In reality neural networks can have singular Fisher Information Matrix. As a result we can assume there is no noise in the data and then project only onto the top subspace of the Fisher Information. This is the same as removing nuisance parameters from linear regression (meaningless parameters with infinite variance as a result). Practically using the Fisher Information as the metric tensor can be singular when we take flat directions into account.

The issue is only relevant for the determinant calculation that we can then do over the top subspace without losing the equivalence to linear regression.

3.7 Equivalence to Maximum A Posteriori

We conclude the three-part proof in this section by showing that the parametrization from the Bayesian approach with the posterior distribution $P(\theta|\mathbf{X}, \theta^*)$ is the same as the parametrization from the MAP solution shown in Section 3.5 with $P(\mathbf{X}, \theta^*|\theta)$. To achieve this we use the Jeffreys prior as, for the exponential family of distributions, it equates the MAP and Bayesian solutions [Hartigan 1998]. This section's lemma is:

Lemma 3: Assuming a Gaussian likelihood and hyper-parameter distribution, and constant second derivatives the Bayesian posterior distribution with a Jeffreys prior is equivalent to Metric Regularization.

We begin with the previously derived integral of our posterior distribution, with the Gaussian likelihood and hyper-parameter distributions with the Jeffreys prior.

$$\int P(\theta|\mathbf{X}, \theta^*)d\theta = \int \frac{1}{Z(2\pi)^{\frac{N}{2}}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{P}(y_i - f(x_i, \theta))^2 - \frac{1}{2}\theta^T I^*(\theta)\theta\right)\left(\sqrt{|I^*(\theta)|}\right)\left(\sqrt{|I(\theta)|}\right)d\theta \tag{3.6}$$

$$\int P(\theta|\mathbf{X}, \theta^*)d\theta = \frac{1}{Z(2\pi)^{\frac{N}{2}}} \int \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{P}(y_i - f(x_i, \theta))^2 - \frac{1}{2}\theta^T I^*(\theta)\theta + \frac{1}{2}\log(|I^*(\theta)|) + \frac{1}{2}\log(|I(\theta)|)\right)d\theta$$

$$\int P(\theta|\mathbf{X}, \theta^*)d\theta = \frac{1}{Z(2\pi)^{\frac{N}{2}}} \int \exp\left(-L(\theta)\right)d\theta$$

We assume we have enough data that the data-to-parameter ratio is at the high dimensional limit: $P/N = \mathcal{O}(1)$

In this limit it is justified to perform the Laplace approximation [Zhang *et al.* 2018],

where it is assumed the model is at a local minimum of parameter space: $\hat{\theta}$

such that $\nabla_\theta L(\hat{\theta}) = \mathbf{0}$ and $\nabla_\theta \nabla_\theta L(\theta) = H(\theta)$ the Hessian Matrix.

$$\int P(\theta|\mathbf{X}, \theta^*)d\theta \approx \frac{1}{Z(2\pi)^{\frac{N}{2}}} \int \exp\left(-L(\hat{\theta}) - \frac{1}{2}(\theta - \hat{\theta})^T H(\hat{\theta})(\theta - \hat{\theta})\right)d\theta$$

$$\int P(\theta|\mathbf{X}, \theta^*)d\theta \approx \frac{1}{Z(2\pi)^{\frac{N}{2}}} \exp\left(-L(\hat{\theta})\right) \int \exp\left(-\frac{1}{2}(\theta - \hat{\theta})^T H(\hat{\theta})(\theta - \hat{\theta})\right)d\theta$$

$\int \exp\left(-\frac{1}{2}(\theta - \hat{\theta})^T H(\hat{\theta})(\theta - \hat{\theta})\right)d\theta$ is a Gaussian integral and so

$$\int \exp\left(-\frac{1}{2}(\theta - \hat{\theta})^T H(\hat{\theta})(\theta - \hat{\theta})\right)d\theta = \frac{(2\pi)^{\frac{N}{2}}}{\sqrt{|H(\theta)|}}$$

This means that $\int P(\theta|\mathbf{X},\theta^*)d\theta \approx \frac{(2\pi)^{\frac{N}{2}}}{Z(2\pi)^{\frac{N}{2}}} \frac{\exp\left(-L(\hat{\theta})\right)}{\sqrt{|H(\theta)|}}$

$$\int P(\theta|\mathbf{X},\theta^*)d\theta \approx \frac{1}{Z} \frac{\exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{P}(y_i - f(x_i,\hat{\theta}))^2 - \frac{1}{2}\hat{\theta}^T I^*(\hat{\theta})\hat{\theta} + \frac{1}{2}\log(|I^*(\theta)|) + \frac{1}{2}\log(|I(\theta)|)\right)}{\sqrt{|H(\hat{\theta})|}}$$

We also see from Section 3.4 that $H(\hat{\theta}) = \nabla_\theta \nabla_\theta L(\hat{\theta}) = I(\hat{\theta})$. Thus:

$$\int P(\theta|\mathbf{X},\theta^*)d\theta = \frac{1}{Z} \frac{\exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{P}(y_i - f(x_i,\hat{\theta}))^2 - \frac{1}{2}\hat{\theta}^T I^*(\hat{\theta})\hat{\theta} + \frac{1}{2}\log(|I^*(\theta)|)\right)\left(\sqrt{|I(\hat{\theta})|}\right)}{\sqrt{|I(\hat{\theta})|}}$$

$$\int P(\theta|\mathbf{X},\theta^*)d\theta = \frac{1}{Z}\exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{P}(y_i - f(x_i,\hat{\theta}))^2 - \frac{1}{2}\hat{\theta}^T I^*(\hat{\theta})\hat{\theta} + \frac{1}{2}\log(|I^*(\theta)|)\right)$$

The final equation above is proportional to Equation 3.4 from Section 3.5 with $\theta = \hat{\theta}$:

$$\frac{1}{Z}\exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{P}(y_i - f(x_i,\hat{\theta}))^2 - \frac{1}{2}\hat{\theta}^T I^*(\hat{\theta})\hat{\theta} + \frac{1}{2}\log(|I^*(\theta)|)\right) \propto$$
$$\frac{1}{Z(2\pi)^{\frac{N}{2}}}\exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{P}(y_i - f(x_i,\theta))^2 - \frac{1}{2}\theta^T I^*(\theta)\theta + \frac{1}{2}\log(|I^*(\theta)|)\right)$$

Since $\hat{\theta}$ is a maximum of Equation 3.4 with a gradient of $0, \hat{\theta}$ is a local minimum of Equation 3.5, our proposed error function with the new regularizer:

$$L(\theta) = \frac{1}{2\sigma^2}\sum_{i=1}^{P}(y_i - f(x_i,\theta))^2 + \frac{1}{2}\theta^T I^*(\theta)\theta - \frac{1}{2}\log(|I^*(\theta)|)$$

This shows that the stable point of our proposed error function (Lemma 1) is the same parametrization as the parametrization found from the full Bayesian solution in Equation 3.6. In Section 3.6 we also showed that the full Bayesian solution is equivalent to the MMSE for linear regression (Lemma 2). Thus, the stable point of our proposed error function is equivalent to the MMSE estimator for linear regression. This concludes the proof of Theorem 1 using Lemma 1, Lemma 2 and Lemma 3.

3.8 Computing Multiplication of Hessian by a Vector

A key component of this work is ensuring that Metric regularization is a feasible method to use for large NNs. This is not trivial, since calculating the Fisher Information Matrix has a memory and time complexity of $\mathcal{O}(N^2)$. We use two identities to make Metric regularization feasible. We use the fact that the Fisher Information Matrix is equal to the Expected Hessian matrix at a minimum of the loss landscape, as discussed in Section 2.4. It is preferable in this work to use the Expected Hessian matrix as we can use the Leibniz Integration rule in Equation 2.4 (the second identity) to change the order of integration and differentiation. This is useful when we are multiplying the Hessian by a constant vector as it allows us to perform the multiplication before the second differentiation and so we never have to store

an $N \times N$ matrix in memory. We demonstrate this below.

$$
\begin{aligned}
H(\theta) &= \nabla_\theta \nabla_\theta L \in \mathbb{R}^{N \times N} \text{ and } V \in \mathbb{R}^N \text{ is some vector independent of } \theta \\
H(\theta)V &= \nabla_\theta \nabla_\theta L V \\
&= V^T \nabla_\theta \nabla_\theta L \\
&= \nabla_\theta (V^T \nabla_\theta L) \text{ since } V \text{ is independent of } \theta
\end{aligned}
$$

Likewise for the expected Hessian:

$$
\begin{aligned}
E_{\mathbf{X}}[H(\theta)] &= E_{\mathbf{X}}[\nabla_\theta \nabla_\theta L] \in \mathbb{R}^{N \times N} \text{ and } V \in \mathbb{R}^N \text{ is some vector independent of } \theta \\
E_{\mathbf{X}}[H(\theta)]V &= E_{\mathbf{X}}[\nabla_\theta \nabla_\theta L]V \\
&= V^T E_{\mathbf{X}}[\nabla_\theta \nabla_\theta L]
\end{aligned}
$$

We can now use the Leibniz Integration rule (Equation 2.4) to change the order of integration and differentiation:

$$= \nabla_\theta (V^T \nabla_\theta E_{\mathbf{X}}[L]) \text{ since } V \text{ is independent of } \theta$$

Note that V and $\nabla_\theta E_{\mathbf{X}}[L]$ are N-dimensional vectors and the dot-product $V^T \nabla_\theta E_{\mathbf{X}}[L]$ results in a scalar value before the second derivative is applied.

This can be used to calculate: $\nabla_\theta(\theta^T H(\theta)\theta) = \theta^T H(\theta)$ in the Metric regularizer if we let $V = \bar{\theta}$ where $\bar{\theta}$ is the values of our parameters that are not tracked by automatic differentiation and we calculate $\bar{\theta}^T H(\theta)$ directly. Since the calculation of the Metric regularizer requires the calculation of the gradient term:$\nabla_\theta(\bar{\theta}^T \nabla_\theta E_{\mathbf{X}}[L])$ we note that at no point does the memory complexity of this calculation grow passed: $\mathcal{O}(N)$

3.9 Simultaneous Power Method using Hessian Multiplication

The simultaneous power method is a computational method for finding the top-k Eigenvectors and Eigenvalues of a matrix. Observing the top Eigenvalues of the Hessian of an NN is a useful way to gain insight into the generalizability of the NN as it has been observe that model parametrizations with smaller Eigenvalues generalize better [Zhang *et al.* 2018]. The computation of the Eigen-spectrum directly from the Hessian for any reasonably large networks, however, is computationally infeasible due to the memory complexity of storing the full Hessian matrix. Thus, Algorithm 1 presents a novel modification of the simultaneous power method which replaces the multiplication between the Q and $\nabla_\theta \nabla_\theta(L)$ matrices, where Q is initialized to a random $N \times k$ matrix. Instead we use the memory efficient Hessian-Vector multiplication from Section 3.8 on each column of Q individually. The rest of the algorithm proceeds as normal, with the following step being to compute the Q,R decomposition of the resulting matrix. Using this modified simultaneous power method now allows us to compute the Eigen-spectrum of realistically sized networks and we use this algorithm to interpret our results in Chapter 5.

Algorithm 1 Simultaneous Power Iteration using Memory Efficient Hessian Multiplication

Require: $\nabla_\theta(L)$ the derivatives of the loss function (L) with respect to the model parameters θ.

Require: k the number of Eigenvalues to be returned

Returns: the top-k Eigenvalues of the Hessian Matrix $\nabla_\theta \nabla_\theta(L)$

procedure $simultaneous_power_iteration(\nabla_\theta(L), k)$
  $n \leftarrow \nabla_\theta(L).dimensions$
  $Q \leftarrow RandomMatrix(n, k)$ ▷ Randomly initializes an $N \times k$ matrix
  $Q, _ \leftarrow QR(Q)$ ▷ Performs the QR decomposition on Q
  $Q_{prev} \leftarrow Q$
  `error` $= 1.0$
  while `error` $> 1 \times 10^{-3}$ **do**
    $Z \leftarrow RandomMatrix(n, k)$
    for $j = 0 \rightarrow k$ **do** ▷ Loops over each column of Q separately
      $q \leftarrow \nabla_\theta(L)^T Q[:, j]$ ▷ compute column k of Q multiplied by the Hessian
      $Z[:, j] \leftarrow \nabla_\theta q$ ▷ using memory efficient Hessian multiplication step
    end for
    $Q, R \leftarrow QR(Z)$
    `error` $= Sum(Q - Q_{prev})^2$
  end while
  return $Diag(R)$ ▷ Returns the diagonal elements of R
end procedure

3.10 Conclusion

In conclusion we demonstrate the following theoretical results in this chapter. Firstly, in Section 3.2, we justify the need for creating algorithms that account for the correlation between parameters by showing that parameters in separate layers of an NN will always be correlated. This means that the particular value for a parameter can only be determined in the context of the rest of the network. We then reflect that Metric regularization, by accounting for the correlation between parameters, is able to achieve the MMSE parametrization for an NN. This is the primary theoretical result of this work with the majority of the sections in this chapter providing pieces of this larger result. One problem with methods that account for the correlation between parameters is that the complexity of this task grows exponentially with the number of parameters in the model. With large NNs this makes these methods restrictive. Thus, Sections 3.8 provides a method of calculating the gradient of Metric regularization which grows linearly with the number of parameters. Thus, we have shown that the independence assumption often used by regularization methods is unjustified and provided a practical alternative, in Metric regularization, which provably benefits from accounting for the correlation between parameters.

Chapter 4

Experimental Methodology

4.1 Introduction

In this chapter we outline the experimental methodology used to evaluate the utility of Metric regularization. We begin in Section 4.2 by describing the five datasets used in this work. These datasets range in difficulty and interpretability, and so provide a balance between insight into the dynamics of the regularizer and realism in the kinds of supervised learning tasks NNs are commonly used for. Supervised learning tasks are either classification or regression tasks and we evaluate both in this work to investigate if our novel regularization methods are consistent for different tasks, and by extension for different loss functions and network architectures. We continue in Section 4.3 by describing the set of learning rules that we use for our experiments. We evaluate a set of different learning rules to gain insight into the strengths and weaknesses of the Metric regularizer as well as to further our understanding of the generalization of NNs. Finally in Section 4.4 we describe our experimental setup as well as the NN architectures used for each learning task.

4.2 Datasets

4.2.1 XOR and XORD Datasets

The first dataset to be used is the Exclusive-Or (XOR) dataset. This dataset consists of 4 data points $X = \{(-1,-1),(-1,1),(1,-1),(1,1)\}$ with corresponding labels $Y = \{-1,1,1,-1\}$. The network must learn to only classify data points positively when the input values disagree. This dataset is a classical problem in NN research as it is non-linearly separable. As a result the multi-layer perceptron was introduced to tackle this problem and fit the data [Rumelhart *et al.* 1985], with the use of hidden layers in the model allowing for a non-linear decision boundary to be learned. This dataset, however, is still relevant today as it offers interpretable insight into the features learned by an NN and its ability to generalize [Brutzkus and Globerson 2019]. As done in Brutzkus and Globerson [2019] the input to hidden layer weights connected to the 2-dimensional inputs can be plotted as vectors on a

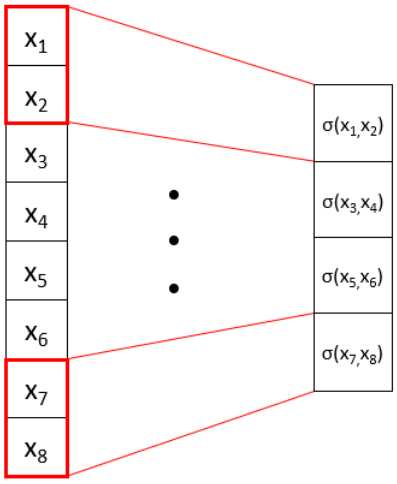


Figure 4.1: Examples of the input-to-hidden convolutional layer for the XORD task. σ is some activation function on the hidden layer.

plane. This provides a useful perspective on the ways that the Metric regularizer differs from the L2 regularizer.

We also use a generalization of the XOR task presented in Brutzkus and Globerson [2019] known as the XOR Detection (XORD) task. In this task the input dimension is $2C$ where C is the number of pairs of input neurons. Each neuron pair is an XOR task on its own, with neurons still having a value of -1 or 1. Instead of using H fully-connected hidden neurons like for the XOR task, we now use H hidden 1-D convolutional layers with a filter size of 2 and step size of 2. The step size of 2 is used so that the filters pass over each input pair individually without an overlap between pairs. An example of this convolutional layer structure is shown in Figure 4.1. The task is then for the network to detect if any of the input pairs has a positive pattern. The hidden to output neurons are fully-connected. We train two networks for this task. One network uses the Tanh activation on its hidden layers, while the other uses the ReLU activation. The convolution filters consist of 2 parameters values and so we can plot these filters in the same manner that we plotted the input to hidden neurons for the XOR task. Similarly this provides some interpretability of the model. Most importantly for the XORD dataset, we can now have an unseen test dataset since we can arbitrarily increase the dimensionality of the input layer by adding more input pairs. In particular we use 5 input pairs resulting in 1024 possible unique input strings of size 10. We generate 12 training samples and 10 test samples to form our dataset. Half of the samples in both sets have a positive label and half have a negative label. All positive samples have at most 1 positive input pair. These settings for generating the dataset were determined empirically and designed to expose overfitting when using vanilla-SGD as the optimization algorithm. None of the regularization techniques were used to determine the dataset parameters to avoid biasing the data in favour of one regularizer.



Figure 4.2: Examples of the MNIST dataset [LeCun *et al.* 2010].

4.2.2 MNIST Datasets

The third dataset used for our experiments is the standard MNIST dataset [LeCun *et al.* 2010]. This dataset is of images of handwritten digits with 60000 training set images and 10000 test set images. Images are grouped into one of the categories $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ reflecting which digit an image represents. An NN must learn to predict the correct label for an image with a one-hot encoding being used as labels over the NNs 10 output neurons. In addition, all input pixel values are normalized to be in the range $[0, 1]$. Figure 4.2 show examples of the MNIST dataset.

4.2.3 Synthetic Dataset

The fourth dataset is a synthetic dataset sampled using the function $y = a \sin(\mathbf{XW}) + a\mathbf{XB} + \varepsilon$ where the inputs and noise are sampled from uniform distributions: $X \sim Uniform(0, b)$ and $\varepsilon \sim \mathcal{N}(0, \hat{\sigma})$. The ground truth weights and biases are also sampled from a uniform distribution: $W, B \sim Uniform(0, c)$. The dimensionality of these variables are $\mathbf{X} \in \mathbb{R}_{+}^{M \times N}$, while $\mathbf{W}, \mathbf{B} \in \mathbb{R}^{N}$, $\varepsilon \in \mathbb{R}^{M}$ and $a, b, c, y, \hat{\sigma} \in \mathbb{R}$. The four parameter values a, b, c and σ are set to control the degree of variance in the data. N and M are the number of data points and input dimensions respectively. Due to the fact that we are generating new data for each experiment we do not have a validation dataset. We use 80% of the N data points for training and 20% as unseen test data (which is used to tune hyper-parameters except for on the final run). For the results shown in Section 5.5 the synthetic dataset is created using the settings of $N = 2000$, $M = 256$, $a = 1.0$, $b = 2.0$, $c = 1.0$, $\hat{\sigma} = 1.0$ resulting in an unconditional variance on y of 29.656715 for the sigmoid dataset and 30.980389 for the ReLU dataset. These hyper-parameters were determined empirically using the SGD based learning rules to obtain a dataset which was challenging but also ensured the models could



Figure 4.3: Examples of the CIFAR-10 dataset [Krizhevsky *et al.* 2009]

overfit to the training data. The two kinds of SGD learning rules are described in Section 4.3. We only used these two learning rules to avoid biasing the results in favour of any of the explicit regularization methods.

4.2.4 CIFAR-10 Dataset

The final dataset used is the CIFAR-10 Dataset [Krizhevsky *et al.* 2009]. This dataset is comprised of 60000 32×32 RGB images, with the input pixel values normalized to be in the range $[0, 1]$. There are 10 classes {airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck} with 6000 images in each class. The dataset is split into 5 training batches totalling 5000 images from each class (some batches may contain more of one class than others) and a final test batch containing 1000 images from each class. Each image contains a single example of any of the 10 possible objects or animals, with a unique label so that there is no instance where two or more possible labels could be appropriate. We use a one-hot encoding to represent the labels for the dataset to be used with a 10-dimensional output layer. Examples of images and their corresponding labels are shown in Figure 4.3.

4.3 Learning Rules

For all of the datasets described above we compare the same set of learning rules summarized in Table 4.1. The first learning rule is SGD on the loss function $L(\mathbf{X}, \theta)$ that provides us with our benchmark. SGD differs from GD only in the fact that it uses a subset of the data to make parameter updates, as opposed to using the full dataset at once. This makes the algorithm more memory efficient. SGD is the most commonly used learning algorithm

for training NNs and has also been shown to reduce overfitting compared to GD [Zhang *et al.* 2018] and so is an appropriate benchmark for this work. The Metric learning rule is the primary rule being investigated and corresponds to the loss function derived and investigated in Chapter 3. This learning rule is SGD on the loss function with regularization based on the Riemannian distance calculated using the Fisher Information Matrix as the metric tensor. The following four learning rules, SGD with L2 regularization and three novel variants of SGD with L2 regularization, are used to compare with the Metric learning rule and gain insight into the strengths of different regularization methods:

- L2 regularization: We derived this learning rule in Section 2.3 and showed that it corresponds to using an independent Multivariate Gaussian as the prior of a Bayesian model. As a result of the independence assumption, the regularizer forces all parameters of a model towards 0 at a fixed regularization rate. As a result it minimizes the L2-norm of the model parameters: $||\theta||_2 = \sqrt{(\theta^T \theta)}$. The L2 learning rule is a useful comparison as it reflects whether minimizing any norm of the model parameters (using Euclidean or Riemannian distance) is sufficient to avoid overfitting and is also widely used in practice.

- Selective L2: This regularization method is of particular interest as it is used to reflect the antithesis of the Metric learning rule. For Selective L2 we only regularize the model parameters that have a small effect on the Riemannian distance. As a consequence only the parameters that are not useful for fitting the training data are regularized. This is in contrast to the Metric learning rule where parameters with a larger Riemannian distance (significant parameters) are regularized more. In Table 4.1 we see that the selection of which parameters to regularize depends on a multiple of the max gradient of the Riemannian distance (the right side of the middle column and fourth row). This condition is evaluated individually for each parameter but the max is taken per layer. This is used to adjust to the different impact of layers in the network but relies on the parameters in a given layer having a similar impact on the Riemannian distance.

- L2 Stopped: This learning rule is used to investigate if L2 regularization can be destructive if applied for too long. To achieve this we use standard L2 regularization for the first τ epochs of training and then switch to using SGD for the remaining epoch. τ is a hyper-parameter of this learning rule and is shown in Table 4.2. In particular this learning rule investigates if minimizing the parameter norm for a reasonable amount of time after initializing in the large regime and then running SGD can recover the performance of running SGD directly from the small regime.

- L2 Stopped Selective: This learning rule investigates a similar trend to L2 Stopped except when L2 regularization is stopped then Selective L2 is used until convergence. This is used to determine if there is benefit to reducing the magnitude of all model parameters before switching to being selective about the regularization. This will reflect if

significant parameters should be moderately regularized before being left to fit the data.

A second consideration in our learning rules is how they relate to the initialization of the model. It has been shown [Geiger *et al.* 2020] that initializing parameters with small weights provides enough of an inductive bias for the model to naturally avoid overfitting, however, initializing with small or large weights is not consistently better or worse for learning the training data. Rather, the benefit of either initialization regime on training data performance is circumstantial and depends on the model architecture and data [Geiger *et al.* 2020]. For brevity we refer to these initialization strategies as the small and large regimes. As a result it is necessary to investigate the SGD and Metric learning rules using both regimes, while the large weight regime is used for the remaining learning rules. This allows us to establish if the small weight regime has enough of an inductive bias to negate the effects of overfitting. This also lets us compare the model weights from the implicit regularization of small weight initialization to enforced small norm solutions from regularizers like L2.

In the learning rules described in Table 4.1 we use the Fisher Information Metric $I(\theta_{(t)})$. This matrix, however, is extremely difficult to compute and hold in memory and a contribution of this work is finding a way to make our regularizers practical. Instead of using the Fisher Information Matrix we used the Expected Hessian matrix, since we can use the efficient Hessian - Vector multiplication from Section 3.8 to calculate all Metric Tensor - Parameter Vector multiplications in the regularizers. Hence we turn a regularizer term, $\theta_{(t)}^T I(\theta_{(t)})$, with compute and memory complexity of $\mathcal{O}(N^2)$ and $\mathcal{O}(N^2)$ respectively into a regularization term, $\nabla_\theta \theta_{(t)}^T \nabla_\theta E_{\mathbf{X}}[L]$, with complexities of $\mathcal{O}(N)$ and $\mathcal{O}(N)$. As shown in Section 3.8, the reason for the efficiency of using the Expected Hessian over the Fisher Information Matrix is because of the Leibniz Integration rule allowing us to change the order of differentiation and integration (averaging over the data points) with the Hessian and then performing the multiplication with the parameter vector before the second differentiation. In practice this θ vector is made constant and not affected by automatic differentiation. Thus, only the derivative of the loss (L) relative to θ is computed. As a result we never hold an $N \times N$ matrix in memory, removing the $\mathcal{O}(N^2)$ memory complexity, or have to perform more than N computations for a single operation, removing the $\mathcal{O}(N^2)$ time complexity. Usually to compute the Hessian matrix the second derivative computes N second derivatives for each of the N first derivatives, and then this would be multiplied by θ to obtain the final values (the same is true for Fisher Information Matrix). By performing $\theta_{(t)}^T \nabla_\theta E_{\mathbf{X}}[L]$ first we reduce the expression to a scalar value, and then the second derivative $\nabla_\theta \theta_{(t)}^T \nabla_\theta E_{\mathbf{X}}[L]$ provides the final N values from this scalar value.

While the use of the Hessian solves the computational issues with the Metric-based regularizers, it is the source of another issue. This is the fact that the Hessian and Fisher Information Matrix are only equal at a critical point in the loss landscape. In addition the Fisher Information Matrix is positive definite everywhere in parameter space, while the Hessian is

Table 4.1: Set of Learning Rules and Initialization Regimes used in the experiments.

Name	Learning Rule	Regime
SGD (Benchmark)	$\theta_{(t+1)} \leftarrow \theta_{(t)} - \alpha \nabla_\theta L(\mathbf{X}, \theta_{(t)})$	Small,Large
Metric	$\theta_{(t+1)} \leftarrow \theta_{(t)} - \alpha \nabla_\theta L(\mathbf{X}, \theta_{(t)}) - \lambda \theta_{(t)}^T H(\theta_{(t)})$	Small,Large
L2	$\theta_{(t+1)} \leftarrow \theta_{(t)} - \alpha \nabla_\theta L(\mathbf{X}, \theta_{(t)}) - \lambda \theta_{(t)}$	Large
Selective L2	$\theta_{(t+1)} \leftarrow \begin{cases} \theta_{(t)} - \alpha \nabla_\theta L(\mathbf{X}, \theta_{(t)}) & \lvert\theta_{(t)}^T H(\theta_{(t)})\rvert \geq \omega \max(\theta_{(t)}^T H(\theta_{(t)})) \\ \theta_{(t)} - \alpha \nabla_\theta L(\mathbf{X}, \theta_{(t)}) - \lambda \theta_{(t)} & \lvert\theta_{(t)}^T H(\theta_{(t)})\rvert < \omega \max(\theta_{(t)}^T H(\theta_{(t)})) \end{cases}$	Large
L2 Stopped	$\theta_{(t+1)} \leftarrow \begin{cases} \theta_{(t)} - \alpha \nabla_\theta L(\mathbf{X}, \theta_{(t)}) & t \geq \tau \\ \theta_{(t)} - \alpha \nabla_\theta L(\mathbf{X}, \theta_{(t)}) - \lambda \theta_{(t)} & t < \tau \end{cases}$	Large
L2 Stopped Selective	$\theta_{(t+1)} \leftarrow \begin{cases} \theta_{(t)} - \alpha \nabla_\theta L(\mathbf{X}, \theta_{(t)}) - \lambda \theta_{(t)} & t < \tau \\ \theta_{(t)} - \alpha \nabla_\theta L(\mathbf{X}, \theta_{(t)}) - \lambda \theta_{(t)} & \lvert\theta_{(t)}^T H(\theta_{(t)})\rvert < \omega \max(\theta_{(t)}^T H(\theta_{(t)})); t > \tau \\ \theta_{(t)} - \alpha \nabla_\theta L(\mathbf{X}, \theta_{(t)}) & \lvert\theta_{(t)}^T H(\theta_{(t)})\rvert > \omega \max(\theta_{(t)}^T H(\theta_{(t)})); t > \tau \end{cases}$	Large

not. Thus, we use the absolute value of the Expected Hessian in our regularizers, which has also been done in prior work with second-order optimization methods [Dauphin *et al.* 2014]. Using the absolute Expected Hessian is justified in our case as it still serves the purpose of reflecting which parameters of the network affect the loss more and can be used to adjust the regularization accordingly. It is necessary to note the loss of precision that occurs anywhere except at a critical point in parameter space and we accept this as a practical consideration of using the Metric-based regularizers. We, thus, now switch our notation from $\theta^T I(\theta)$ to $\theta^T H(\theta)$ where $H(\theta)$ is the Hessian matrix. This does not explicitly reflect the use of our efficient calculation of the derivative, however, it is used in all cases.

4.4 Experimental Setup

The empirical results shown in Chapter 5 are obtained by training the same network architecture repeatedly from different initializations using each learning rule shown in Table 4.1. The architecture, number of training runs and hyper-parameters vary depending on the dataset being used. To ensure that we are getting a fair comparison between the learning rules we ensure that each learning rule is started from the same set of initializations. For example, if we intend to train a network 30 times for each learning rule, then 30 seeds are created and used to initialize each network. This only includes initializations from the same regime, since we are also comparing between the small and large weight initialization regimes. Thus, all weight initializations from the same regime are the same, however, initializations from different regimes are necessarily different. For each learning rule we track the training data accuracy/error, test data accuracy/error and L2 parameter norm at the end of each epoch of training (where accuracy is used for classification tasks and error for regression tasks with

the mean squared error). We then use the repeated training runs to obtain the mean and standard deviations of these metrics at each epoch. In addition, we also calculate the mean and standard deviation of the top values from the Eigen-spectrum of the Expected Hessian matrix of the network parameters at the end of each run.

An important consideration for our experiments is how we set the hyper-parameters for each learning rule and initialization regime. To begin with we optimize hyper-parameters that are common between all learning rules using the training and validation results on vanilla SGD. These hyper-parameters are the learning rate and batch size. Batch size is the same for all training runs, while different initialization regimes have their own optimized learning rate that is the same for all learning rules using that initialization regime. For all datasets except CIFAR-10 we sample the weights uniformly between 0 and a maximum value that is set based on the initialization regime we are aiming to use. This maximum sampling value is also dependent on the network architecture being used and the dataset. For each experiment we empirically determine this value using SGD only, to avoid biasing the results of the different regularizers. Setting this maximum value too small or too large will also break the training procedure altogether. We, thus, found values that enabled the model to fit the training data perfectly but reproduced the generalization gap between the two regimes seen in the literature [Geiger *et al.* 2020]. For CIFAR-10 variance scaling with a Glorot Normal distribution [Glorot and Bengio 2010] was used to initialize the weights where we changed the scaling to find the different initialization regimes.

The learning rule specific hyper-parameters are then optimized in parallel. These include the L2, Metric and Selective L2 regularization rates, the point that regularization was stopped for the epoch based regularizers (τ in Table 4.1) and the selection threshold for the selective regularizers (ω in Table 4.1). The L2, Metric and Selective L2 regularization rates are optimized in a standard fashion using validation data. To ensure that using a fixed learning rate while optimizing these hyper-parameters did not bias the results in favour of one learning rule, we also do a full parameter sweep for the learning rate and regularization rates for MNIST with ReLU activation and the synthetic dataset with Tanh activation. Naturally this kind of approach is note feasible due to the number of experiments being run. However, the results from both hyper-parameter optimization approaches agree and thus, we are comfortable to proceed with the hyper-parameter optimization as described above for the experiments. For the stopping point of the epoch based regularizers, we set this value as any epoch after which a large regime network could have a similar parameter L2-norm, $||\theta||_2 = \sqrt{(\theta^T\theta)}$, to an SGD model from the small regime. The purpose of these regularizers was to identify if indiscriminately regularizing the weights into the small regime and then stopping regularization would match training with SGD from the small regime. Lastly the selection threshold on the selective regularizers was optimized using validation accuracy as well as the parameter norms. Qualitatively, if the parameter norm decreased too significantly past the small initial weight regime norm then we decrease the threshold to be less

selective about which parameters are important to the network (and so fewer parameters get regularized). If the parameter norm decreased too little then we increase the threshold so that we regularize more parameters.

Another necessary consideration around the generality of our results is the activation functions used in the network architectures. It has been observed that changing the activation function of a network can have a significant impact on the performance of the network. Thus, it is also necessary for us to verify all of the regularizers on different activation functions. We examine two different activation functions, the hyperbolic-tangent (Tanh) and Rectified Linear Unit (ReLU). We decided on these two functions as they are representative of two different types of activations, sigmoidal shaped activations and ReLU based activations (in recent work there have been multiple variants of the ReLU activation created [Maas *et al.* 2013; Clevert *et al.* 2015; Klambauer *et al.* 2017]). For classification tasks we use the Softmax function on the output layer with the cross-entropy loss and for regression tasks we use a linear output layer with quadratic loss. The datasets, final hyper-parameters and type of architecture used for the experiments are summarized in Table 4.2. Table 4.2 also links each experiment to a diagram of the architecture used.

Figure 4.4 shows the architecture used for the XOR task. This is a simple fully-connected network with one hidden layer. The simplicity of this network allows us to interpret the parameters learned during the XOR task which is useful to understand our various regularizers. Figure 4.5 shows the architecture for the XORD task. The motivation for this network is again simplicity, however, in this case the one hidden layer is convolutional. For the MNIST task we use the network shown in Figure 4.6. This network has two hidden fully-connected layers and is the same network used in Geiger *et al.* [2020] to analyse the small and large initialization regimes. The network for the synthetic regression dataset shown in Figure 4.7 is similar with two fully-connected hidden layers. For both of the network architectures used on the MNIST and synthetic datasets we use reasonably sized networks on more difficult tasks to start challenging the regularizers, however, the architectures are able to completely fit the training data without more complicated layers such as convolutional layers. Lastly Figure 4.8 shows the architecture used for the CIFAR-10 dataset. This is the most complicated network and is used to show the effect of our regularizers on a large network, commonly used in practice with various layers types and on a complicated dataset. We believe that these architectures and datasets provide an incremental increase in difficulty while providing the best combination of interpretability and practicality.

Finally, to obtain the top Eigenvalues of the network parameters at the end of training for each model we use Algorithm 1 from Section 3.9. This algorithm is a novel version of simultaneous power iteration designed to find the top Eigenvalues of the Hessian of an NN. As a result we are able to find the Eigenvalues for realistically sized networks in an efficient manner, something that previously was impossible without multiple GPUs. In our case a

Table 4.2: Hyper-parameters used in each experiment.

	XOR 4.4		XORD 4.5		MNIST 4.6		Synthetic 4.7		CIFAR-10 4.8	
Activation	Tanh	ReLU	Tanh	ReLU	Tanh	ReLU	Tanh	ReLU	Tanh	ReLU
Learning Rate (Small)	0.12	0.12	0.02	0.05	0.001	0.001	0.0002	0.0002	0.005	0.005
Learning Rate (Large)	0.0015	0.002	0.02	0.002	0.001	0.001	1e-4	6e-5	0.005	0.005
Metric Reg Rate	0.008	1e-4	1e-4	0.001	1e-4	1e-4	5e-4	2e-6	1e-5	1e-5
L2 Reg Rate	0.001	0.001	1e-5	1e-4	1e-5	1e-5	1e-4	1e-5	1e-5	1e-5
Selective L2 Reg Rate	0.1	0.1	0.001	0.001	0.001	1e-5	0.1	0.001	5e-5	5e-5
τ	280	280	500	500	280	280	600	600	490	490
ω	0.1	0.05	0.01	0.1	0.001	0.01	0.0005	0.005	0.02	0.02
Init σ^2 (Small)	0.03	0.03	0.4	0.2	0.1	0.1	0.05	0.02	0.2	0.2
Init σ^2 (Large)	2.3	1.7	4.0	2.0	2.0	2.0	0.5	0.2	5.0	5.0

single GPU is used. This now affords us the opportunity to empirically evaluate whether smaller Eigenvalues from the Hessian of the loss function correspond to better generalization as is commonly found in previous literature [Dauphin *et al.* 2014]. This is also another result that we can use to interpret the learning rules in Table 4.1. Since the creation of this algorithm is a contribution of this work, we empirically verify that it is correct in Section 5.2. To determine the number of Eigenvalues to obtain from the simultaneous power method we use the number of output neurons from the model. The only exception is for the synthetic regression dataset where we obtain 3 Eigenvalues even though the model has 1 output neuron. We choose to match the number of output neurons as previous work has shown that the number of significant Eigenvalues in the model matches the number of output neurons [Gur-Ari *et al.* 2018]. For all learning rules we calculate the Hessian relative to the loss portion of the learning rule only to ensure that we are making a fair comparison.

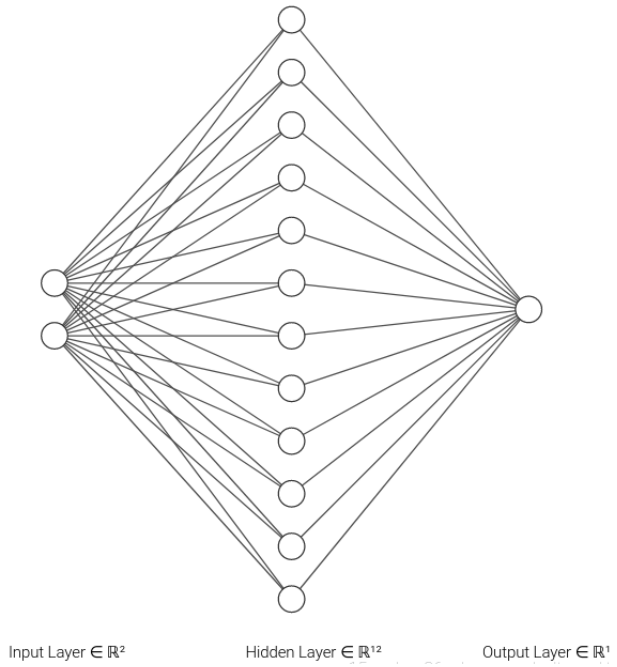


Figure 4.4: XOR Network Architecture

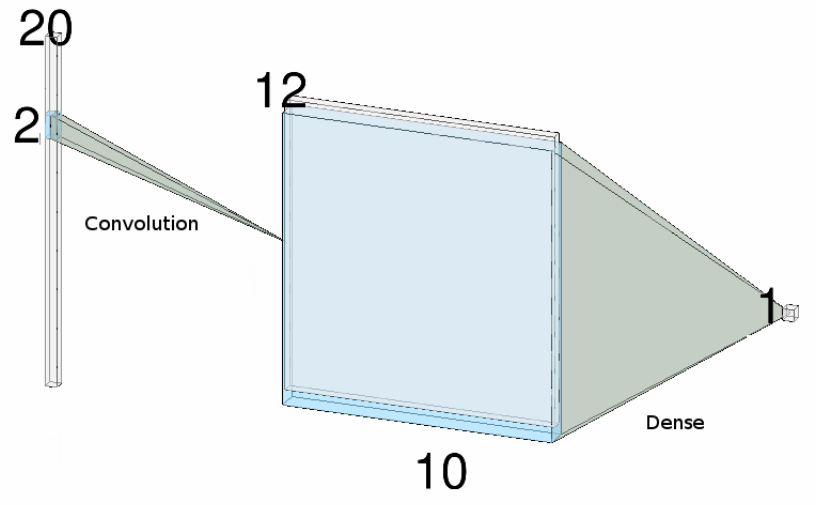


Figure 4.5: XORD Network Architecture

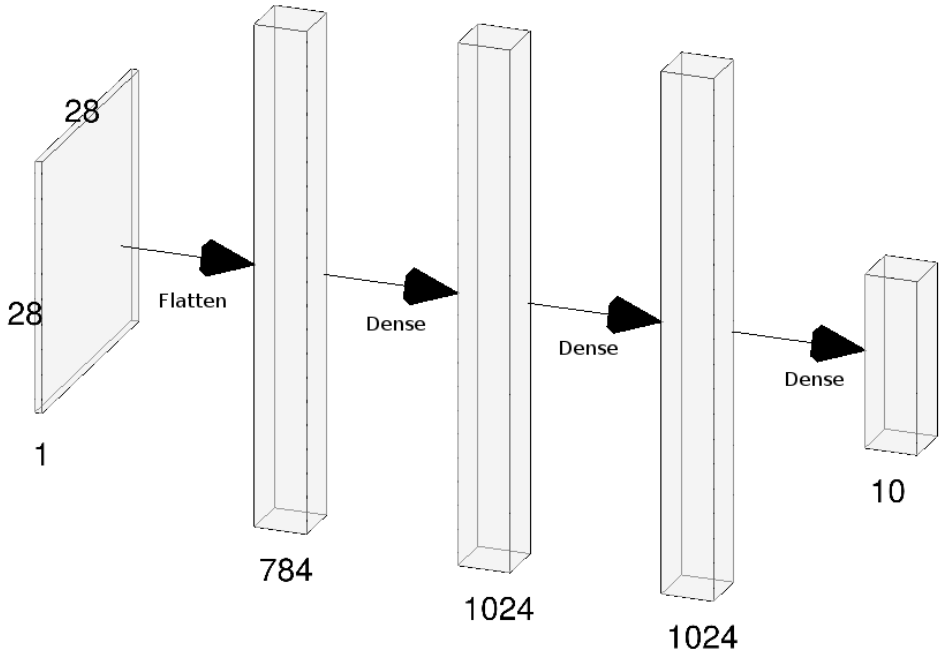


Figure 4.6: MNIST Network Architecture

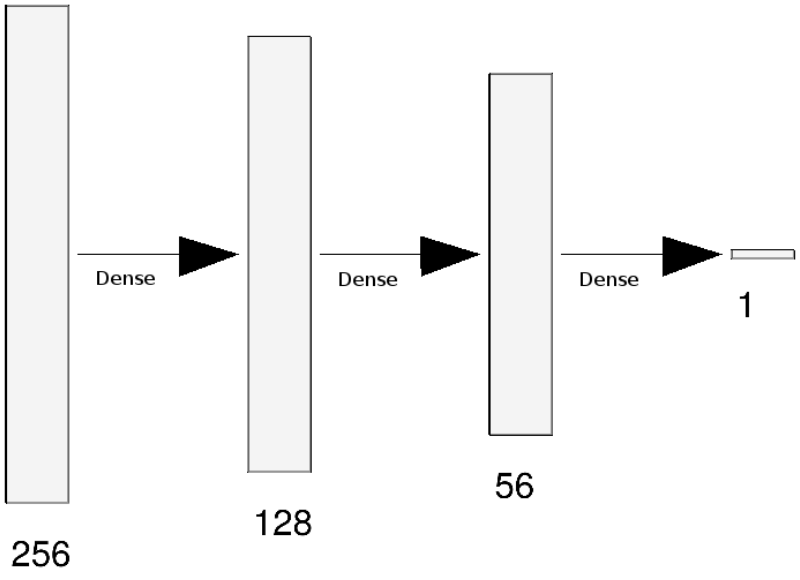


Figure 4.7: Synthetic Data Network Architecture

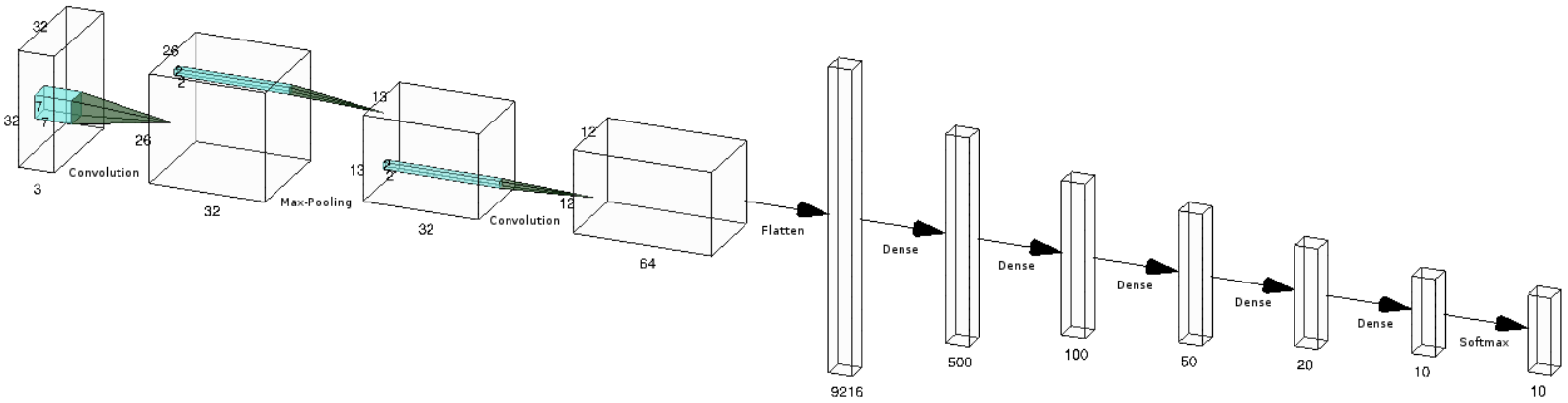


Figure 4.8: CIFAR-10 Network Architecture

Chapter 5

Experimental Results

5.1 Introduction

In this chapter we present the results of the experiments described in Chapter 4. We begin by verifying the correctness of the Efficient Power Method algorithm in Section 5.2. This algorithm is described in Section 3.9 and used to calculate the Eigenvalues of the Hessian matrix from the loss of an NN. The following sections are in the same order as the sections of Chapter 4. We begin with the XOR and XORD datasets' results in Section 5.3. We then follow with the MNIST and Synthetic datasets in Sections 5.4 and 5.5 respectively. We conclude with the results of the CIFAR-10 experiments in Section 5.6.

5.2 Checking Efficient Power Method

In this section we briefly check that the Efficient Power Method algorithm that we outline in Algorithm 1 obtains the same top Eigenvalues from the loss of an NN as performing the (standard) Power Method on the full Hessian matrix of the loss function. To verify this we train a small fully-connected model (as large as we could fit on one GPU while calculating the full Hessian) with layer size $[100, 10, 5]$ to perform a sub-task on the MNIST dataset. The architecture is shown in Figure 5.1. For this task we extract the center 10×10 square from each image and flatten it as input to the model. To ensure that this small network could still fit the data we decrease the difficulty by sampling images and labels exclusively from the first 5 digits and the model aims to classify this subset of the MNIST data. At the end of training we calculate the top 15 Eigenvalues using both methods outlined above. Figure 5.2 shows the resulting two sets of Eigenvalues. We see that the two methods match exactly. Thus, we may use the Efficient Power Method to accurately obtain the Eigenvalues of the Hessian matrix for realistically sized model, and we do so to gain insight into the generalizability of these models.

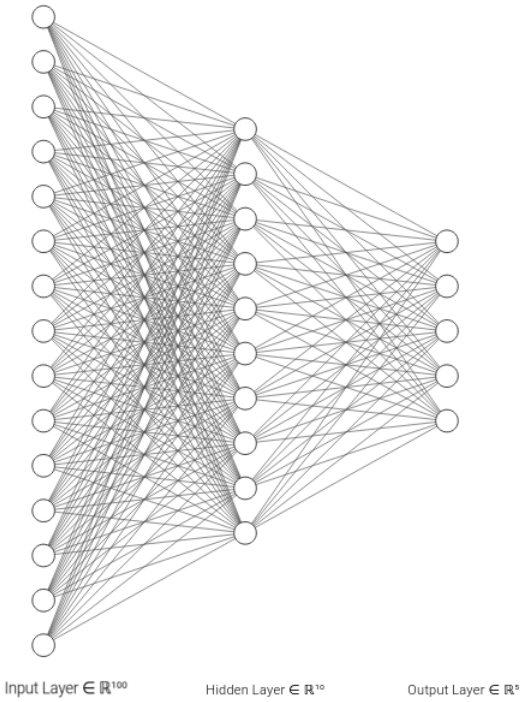


Figure 5.1: Network Architecture used to verify that the Efficient Power Method algorithm provides the correct output.

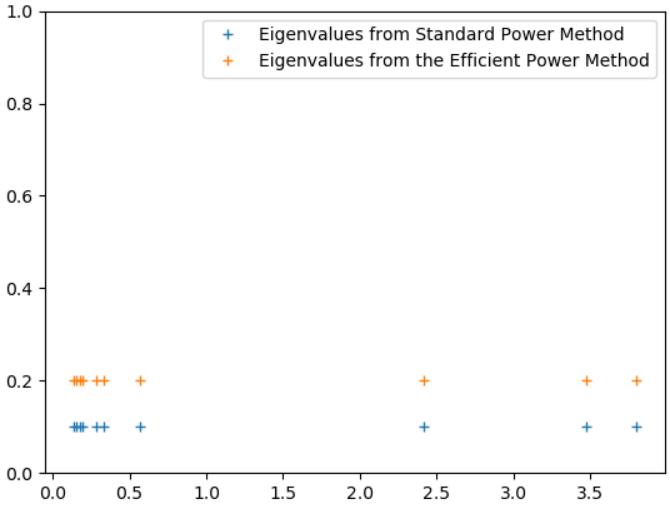


Figure 5.2: Top 15 Eigenvalue from the Standard Power Method on the full Hessian of the loss (blue) compared to the Eigenvalues determined by the Efficient Power Method from Algorithm 1 (orange). Distance (using square root of $L2$-norm) between Eigenvalues from Standard vs Efficient Power Method: 0.00002146.

5.3 XOR Datasets

The mean training accuracy of the eight learning rules shown in Table 4.1 on the XOR dataset can be see in Figure 5.3 for the Tanh activation. Firstly we note that all learning rules converge to a mean accuracy of roughly 100% by the end of training. There is some variance towards the end of training for the *Metric Small Regime* and *Metric Large Regime* updates. For the large regime *Metric* learning rules the variance is constant over multiple epochs, indicating that there were some training runs where the accuracy never reached 100%. For the small regime *Metric* learning rule the variance is non-zero on certain epochs, indicating that the accuracy dropped below 100% momentarily but then returned to perfect accuracy. Further, we note that for this dataset and Tanh activation the large initialization regime trains faster than the small initialization regime. In addition regularization slows down convergence of the models as expected.

In Figures 5.4 we can see that the large initialization regimes maintain a large weight norm throughout training compared to the small initialization regime models. This is significant as practical and theoretical prior work has found that small-norm model parametrizations generalize better than large-norm parameterizations [Bansal *et al.* 2018]. We note that for the large regime, without regularization the model norm stays relatively constant (see *SGD Large Regime*) and even when *L2* regularization is stopped early the weight norm then remains constant (*L2 Stopped*). A further observation is the consistency of the final weight norm for the *Selective L2* and *L2 Stopped Selective* learning rules. This indicates that the *Selective L2* regularizer produces a similar final weight norm regardless of when it is applied or from the initial weights to which it is applied (as long as they are sufficiently large to begin with). It is also apparent that the small initialization regime produces the lowest norm solutions without regularization. From these results we may conclude that all learning rules were able to learn the XOR task to a near perfect level, with training time being the primary difference between the learning rules and initialization regimes.

A benefit of the XOR dataset, as mentioned in Section 4.2.1, is the fact that the input-to-hidden layer weights can be plotted as vectors on a 2-dimensional plane and interpreted. In particular each vector shows that a pair of input-to-hidden parameters (one connected to each input) are responding to a certain feature. For example if a vector is pointing directly at the $(-1, -1)$ input then we know that a pair of parameters is looking particularly for this pattern in the input values. The length of the vector then shows how sensitive the model is to that input pattern, or in other words how confident the model is that the input pattern is helpful for separating the data. These plots are shown in Figure 5.5 for each learning rule at the end of training. From these plots we can see that the small initialization regime results in a minimal set of significant parameters being learned which point directly at two of the input patterns. By contrast we see that the large initialization regime has parameters pointing in a variety of different directions, most of which do not correspond exclusively to

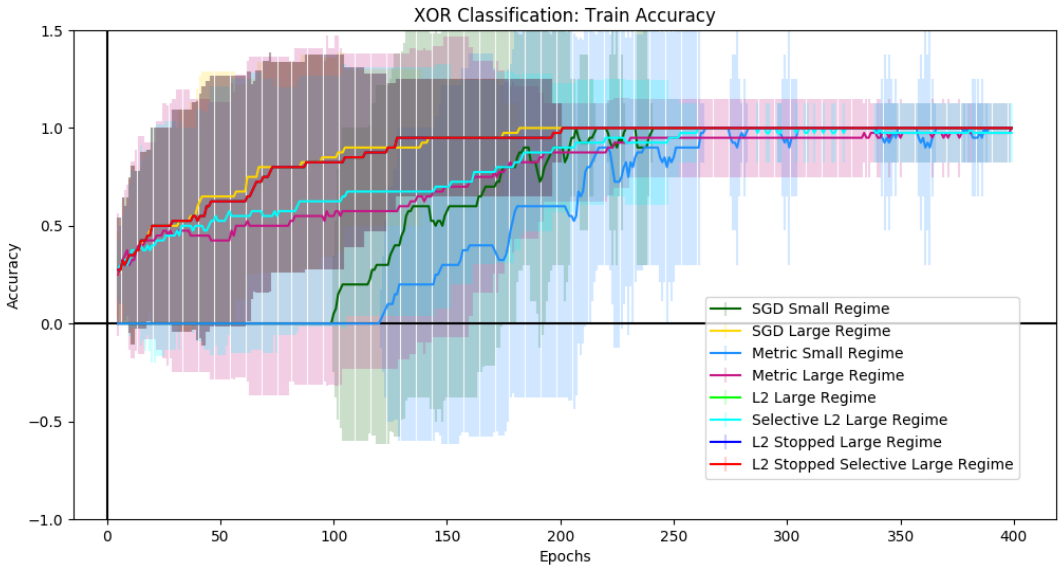


Figure 5.3: Mean Train Accuracy for XOR data with Tanh Activation over 10 runs. Shaded region is 2 standard deviations.

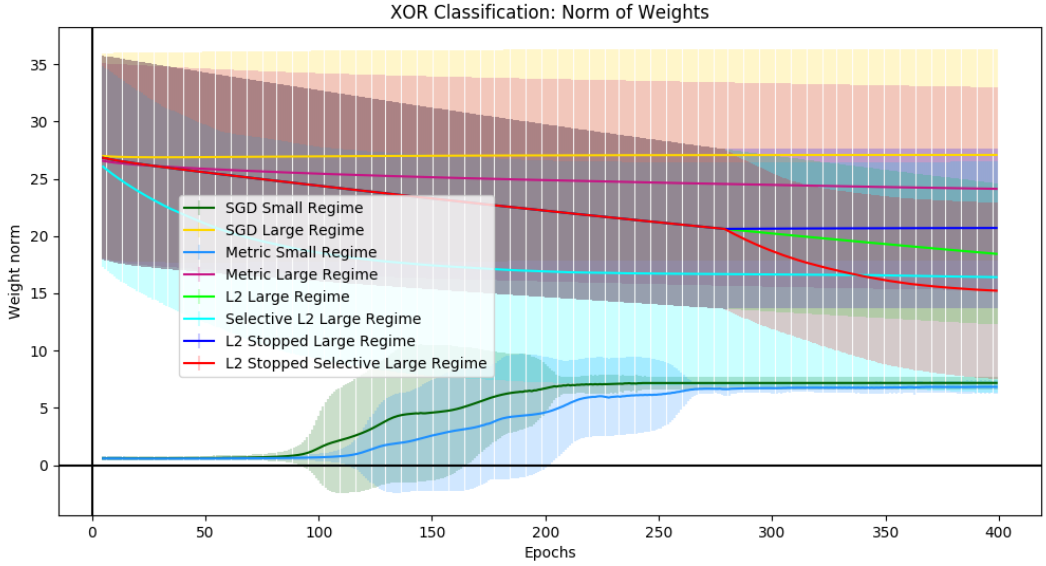


Figure 5.4: Mean Parameter Norms for XOR data with Tanh Activation over 10 runs. Shaded region is 2 standard deviations.

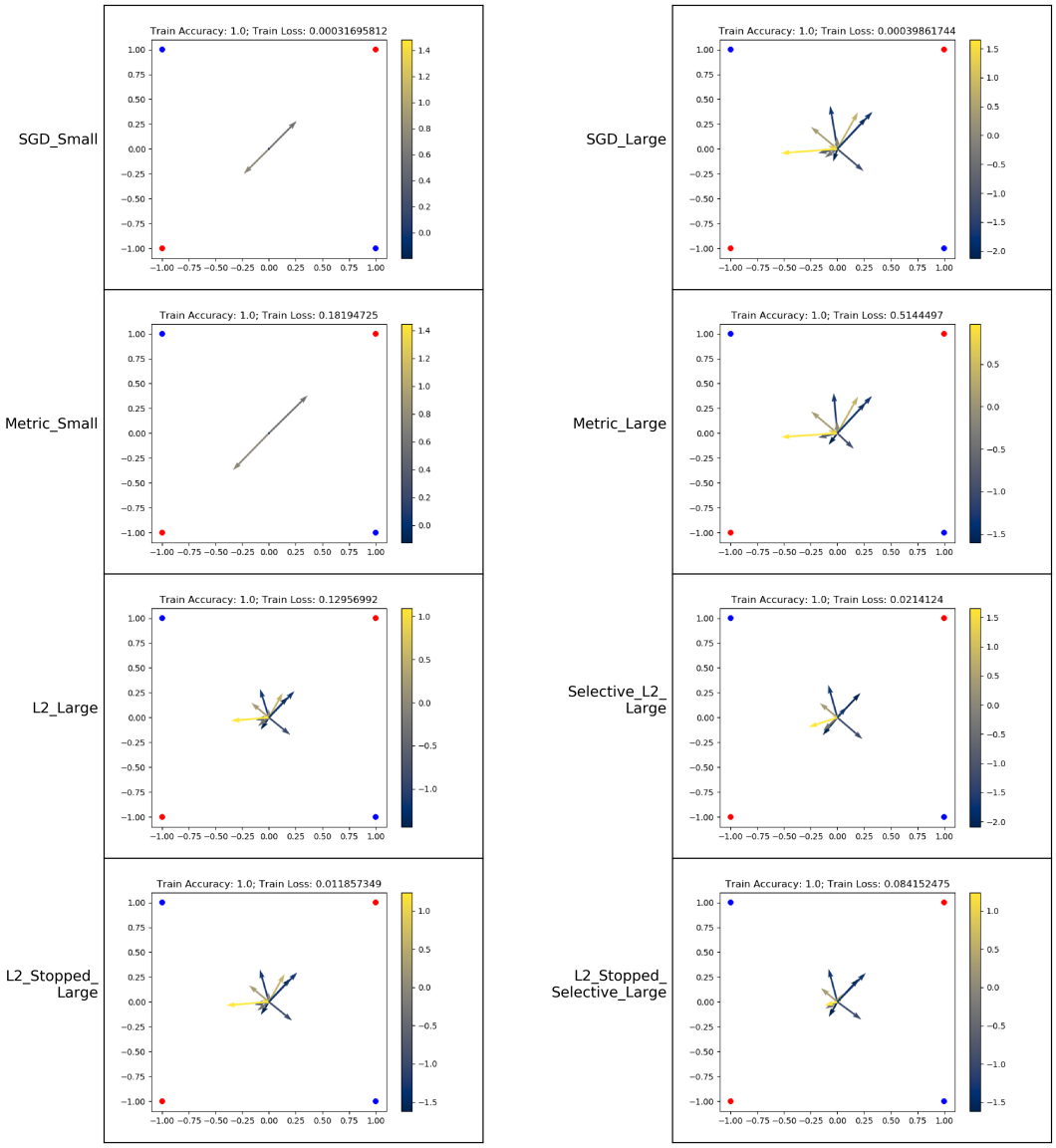


Figure 5.5: Plot of all input-hidden layer parameters from XOR data with Tanh Activation.

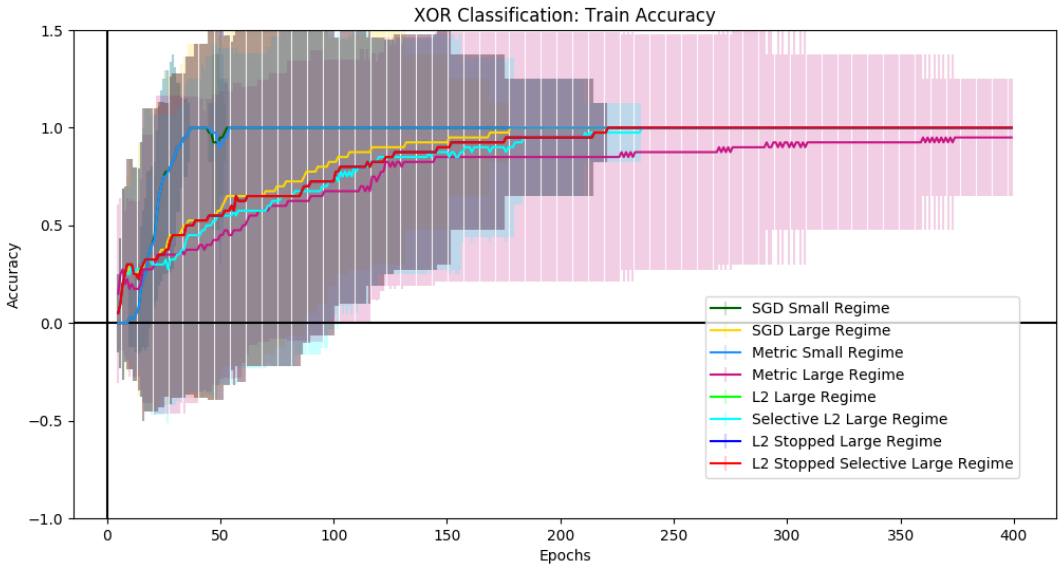


Figure 5.6: Mean Train Accuracy for XOR data with ReLU Activation over 10 runs. Shaded region is 2 standard deviations.

a single input pattern. We also note that for the *SGD*, *Metric*, *L2* and *L2 Stopped* learning rules in the large regime more significant parameters are used than with the *Selective L2* and *L2 Stopped Selective* learning rules in this regime. The effect of *L2* in this case appears to be in reducing the norm of the individual vectors but not in reducing the quantity of significant vectors in the model. While *Selective L2* is effective at reducing the quantity of significant parameters, it still maintains more significant parameters than the small regime models. Yet it is more apparent that each vector corresponds to a certain input pattern than with the other large regime models.

Similar conclusions can be drawn for the XOR dataset when the ReLU activation is used. The primary difference can be seen in Figure 5.6, that the small weight regime trains faster in this case and that the *Metric* learning rule from the large regime was unable to achieve a mean accuracy of 100%. All other learning rules achieve 100% training accuracy with no variance passed epoch 250. Additionally in Figure 5.8 we see that the small weight regime now uses 4 significant vectors each corresponding exactly to one of the input patterns. The *Selective L2* learning rules in this case use only three significant vectors, however, one of the three does not correspond exactly to a single input pattern. It is still apparent that the *Selective L2* regularizer results in fewer significant vectors being used compared to the other large regime models and it more closely resembles the small regime vector plots.

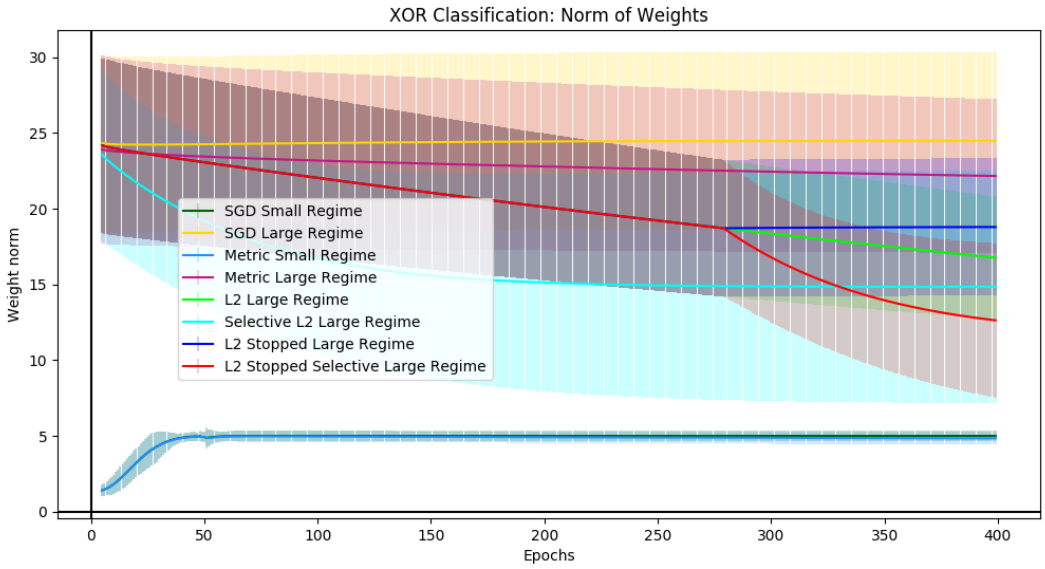


Figure 5.7: Mean Parameter Norms for XOR data with ReLU Activation over 10 runs. Shaded region is 2 standard deviations.

For the XORD dataset we look at the mean squared error as opposed to the accuracy as we found it helped with separating the performance of the models. The mean train error of the model with Tanh activation is shown in Figure 5.9. From these results we can see that all models achieve a near 0 error. The two small initialization regime models and the two *Selective L2* based learning rules are the only learning rules that do not achieve exactly zero on this task. We again see that the large initialization regime models train faster than the small initialization regime models, which is consistent with our findings on the XOR task. Looking at the test error in Figure 5.10, however, we see that the small initialization regime models achieve the lowest test error, followed by the *Selective L2* and *L2 Stopped Selective* learning rules. Thus, the models that achieved the lowest training error also had the worst test accuracy, resembling the traditional notion of overfitting. Counter-intuitively the two *L2* based learning rules generalized the worst on average. Looking at the norms in Figure 5.11 we see that the magnitude of the norms of the weights is a good indication of the generalizability of the model, with the models that do the best also having the lowest norm.

Lastly, we may again plot the input-to-hidden weights of the models in Figure 5.12, this time for the learned convolutional filters. We see that the small initialization regime results in models that use a few significant vectors that clearly correspond to individual input patterns. The contrast with the large initialization regime models is more severe here than with the XOR task. In this case we see that the convolutional filters explode in magnitude and do

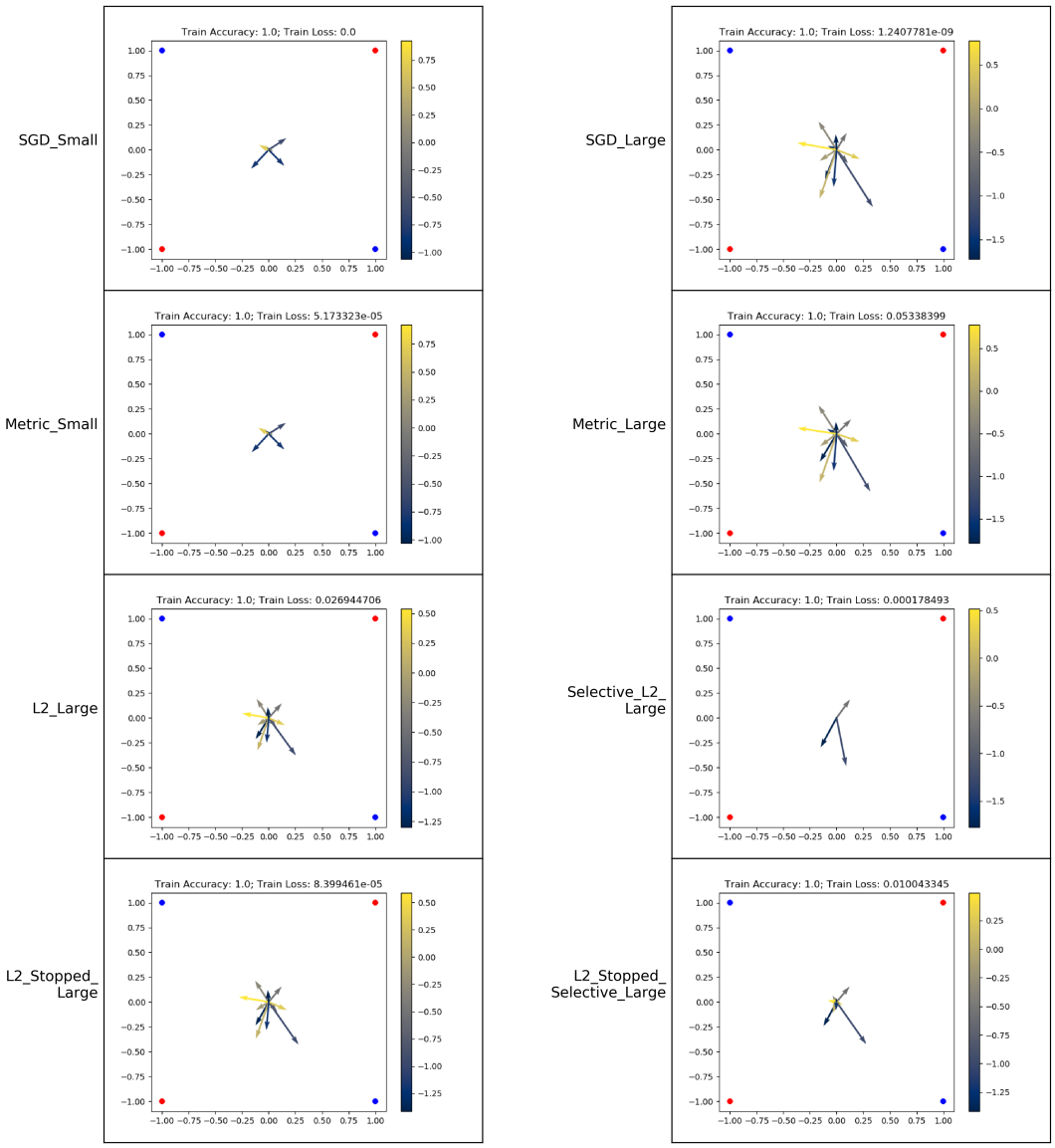


Figure 5.8: Plot of all input-hidden layer parameters from XOR data with ReLU Activation.

not correspond exactly to any particular input pattern. The *Selective L2* filters have the smallest magnitudes of the large regime models and have the least significant vectors that do not point in the direction of one input pattern. The *Metric* regularizers do not help with tightening the filters and appear to have the most spread input filter vectors. *L2*, *L2 Stopped* and *L2 Stopped Selective* appear to result in filters that point more in the direction of particular input patters compared to the *SGD* model from the large regime. The benefit

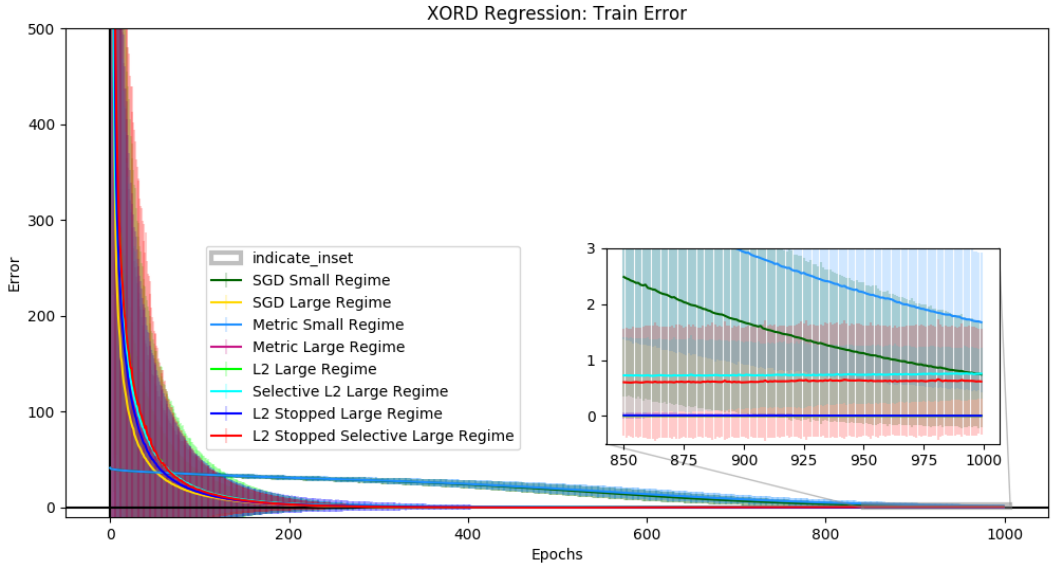


Figure 5.9: Mean Train Errors for XORD data with Tanh Activation over 10 runs. Shaded region is 2 standard deviations.

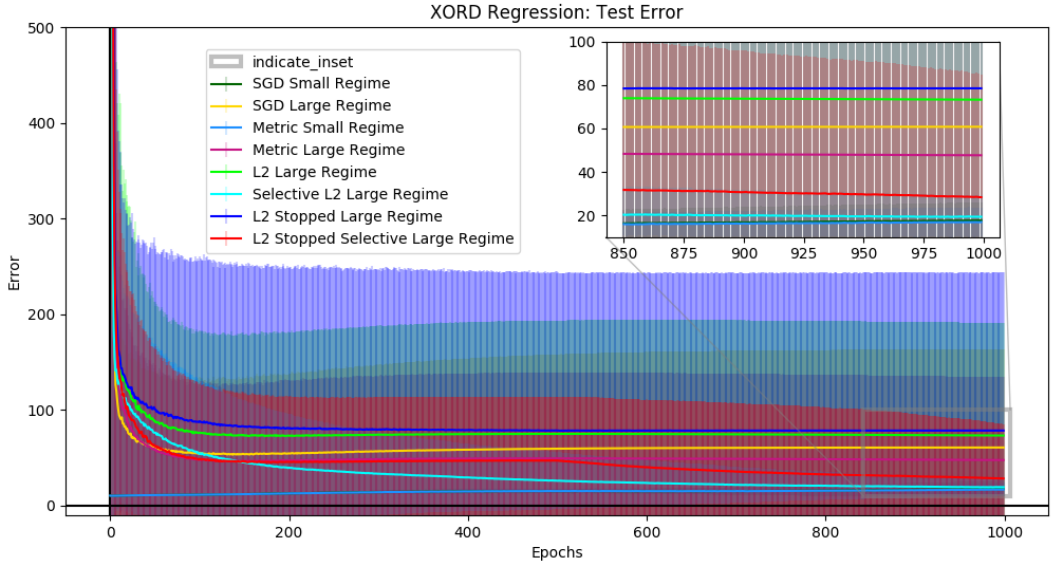


Figure 5.10: Mean Test Errors for XORD data with Tanh Activation over 10 runs. Shaded region is 2 standard deviations.



Figure 5.11: Mean Parameter Norms for XORD data with Tanh Activation over 10 runs. Shaded region is 2 standard deviations.

of the XORD task is that we can see how the pattern of the input filters correlate to the generalizability of the model. From these results it is clear that the models with smaller-magnitude and fewer significant vectors generalize better than the models that have larger filter vectors that are more spread out. It is not surprising then that the *Selective L2* learning rule aids generalizability more than the *Metric* learning rule as *Selective L2* regularizes the insignificant parameters of the model, which are the filters that do not correspond to a single input pattern. In contrast the *Metric* regularizer regularizes the filters that point towards particular input patterns as they are the most useful for reducing the model's loss. This results in filters that are more fanned out, as can be seen in Figure 5.12.

If we analyze the results of the XORD task with the ReLU activation we see that the small regime models achieve 0 training error in Figure 5.13, while the large regime models do not. In particular the *Metric* large regime model obtains a relatively large training error. The relative performance is the same for test accuracy in Figure 5.14. Note that the training error is larger than the test error because there is a larger sample of training examples and we report the sum of the error. It is worth noting that in this case the *Metric* regularizer does appear to offer some benefit to the small regime model's generalizability. The *Metric* regularizer still achieves the worst results from the large regime. The norms of the models again provide an accurate indication of the generalizability of the models in Figure 5.15. Finally, looking at the convolutional filters plotted in Figure 5.16 we see that the small regime results a few significant filters being learned that correspond to particular input patterns.

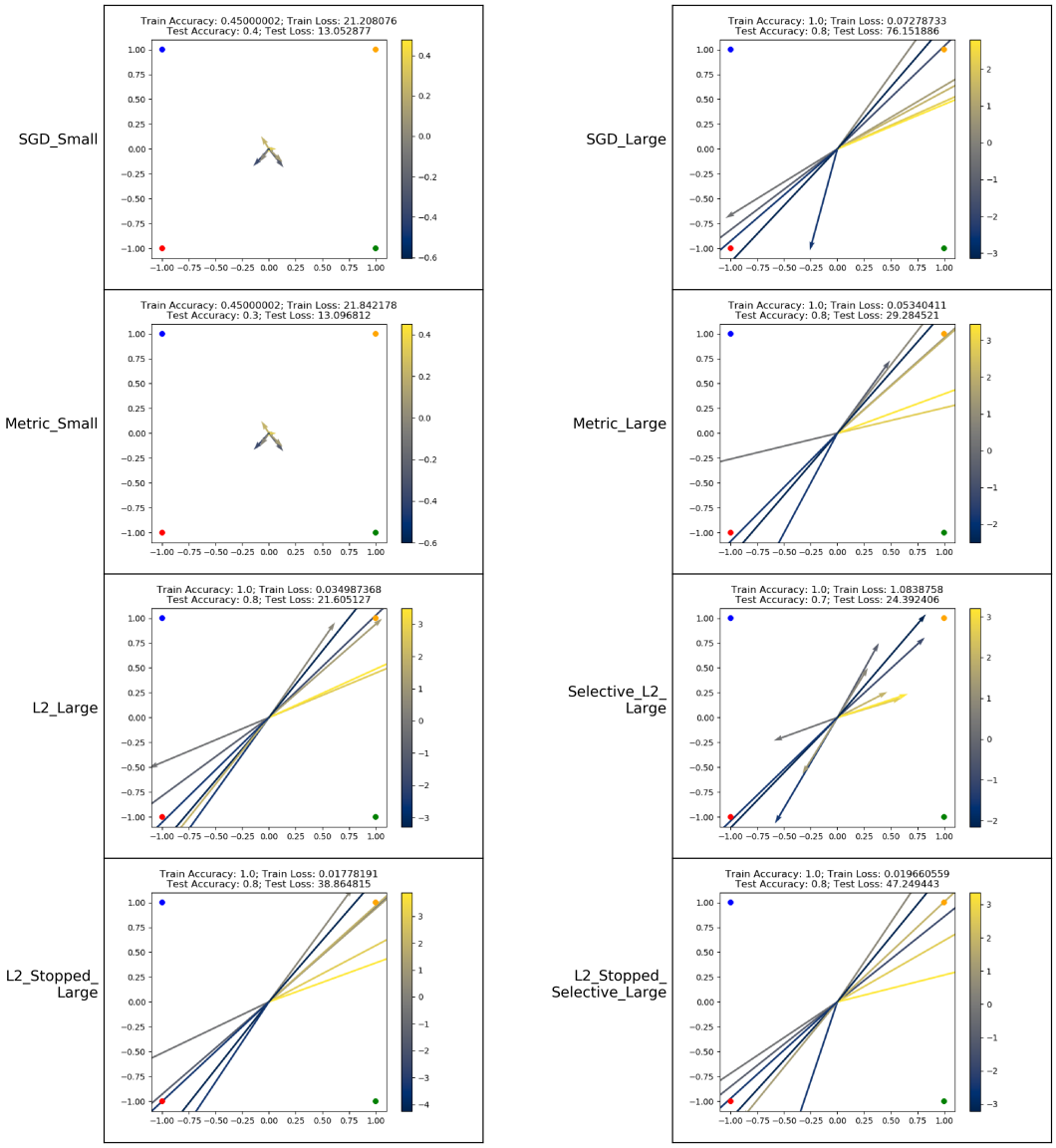


Figure 5.12: Plot of all input-hidden layer parameters from XORD data with Tanh Activation.

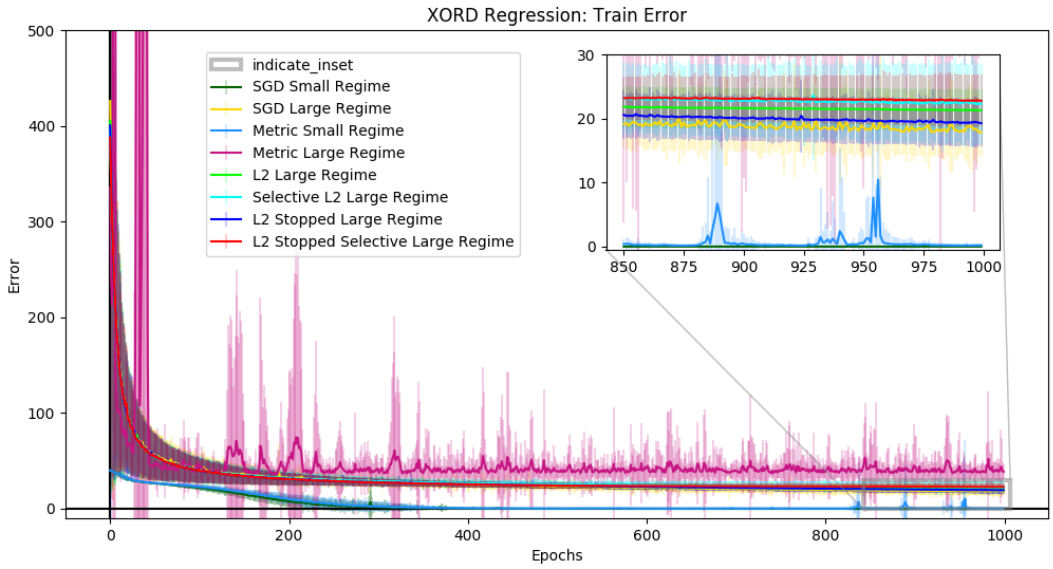


Figure 5.13: Mean Train Errors for XORD data with ReLU Activation over 10 runs. Shaded region is 2 standard deviations.

The large regime models use more filters that do not capture individual input patterns. The magnitudes of these filters are smaller when using the ReLU activation because ReLU does not saturate like the Tanh activation, and initializing the model with larger weights made training unstable.

By empirically analyzing the performance of the learning rules and qualitatively interpreting their final input-to-hidden parameters we can begin to gain some intuition for why there is a discrepancy in the generalizability of the small and large regime models and how to regularize models to replicate and potentially improve upon the implicit regularization of small weight initialization. In particular we see that small weight initialization does more than just limit the model to small-norm solutions but also limits the model to a minimal use of significant parameters that describe the dataset more concisely. While the XOR and XORD datasets are easy tasks, the interpretability that they offer is helpful for us to analyze the results on the more realistic datasets.

5.4 MNIST Datasets

We begin this section by looking at the training accuracy of both the Tanh and ReLU models shown in Figures 5.17 and 5.20 respectively. Firstly we note that most learning rules were able to achieve over 94% mean training accuracy on this dataset. The exceptions to this is *Metric* regularization with Tanh activation which achieves only about 70% accuracy.

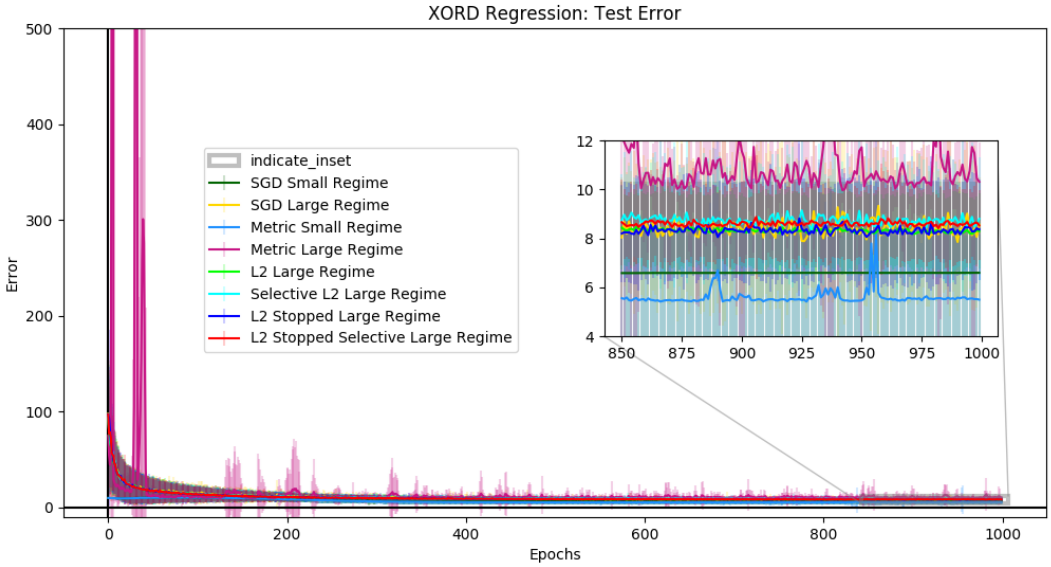


Figure 5.14: Mean Test Errors for XORD data with ReLU Activation over 10 runs. Shaded region is 2 standard deviations.

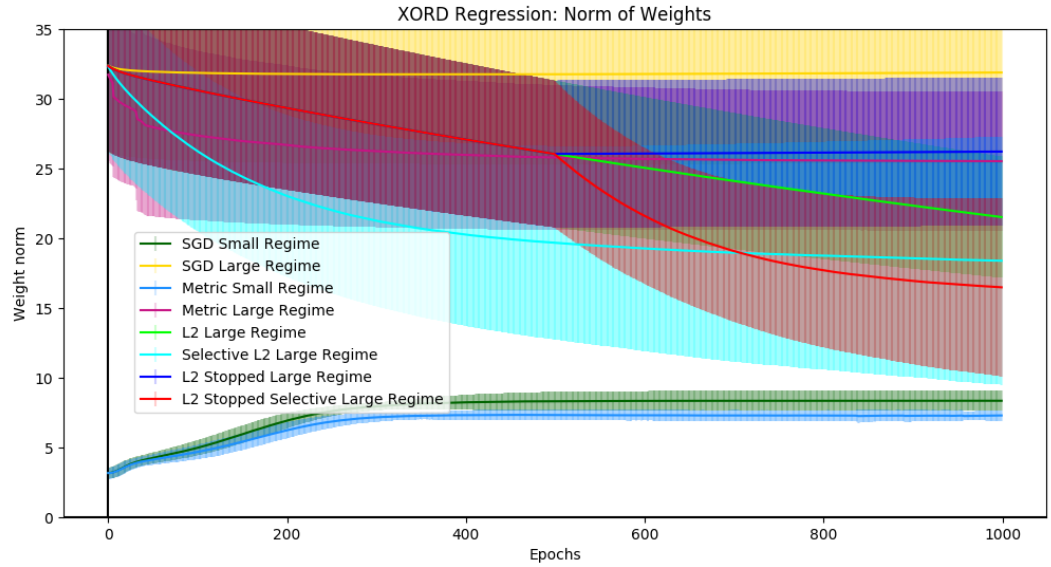


Figure 5.15: Mean Parameter Norms for XORD data with ReLU Activation over 10 runs. Shaded region is 2 standard deviations.

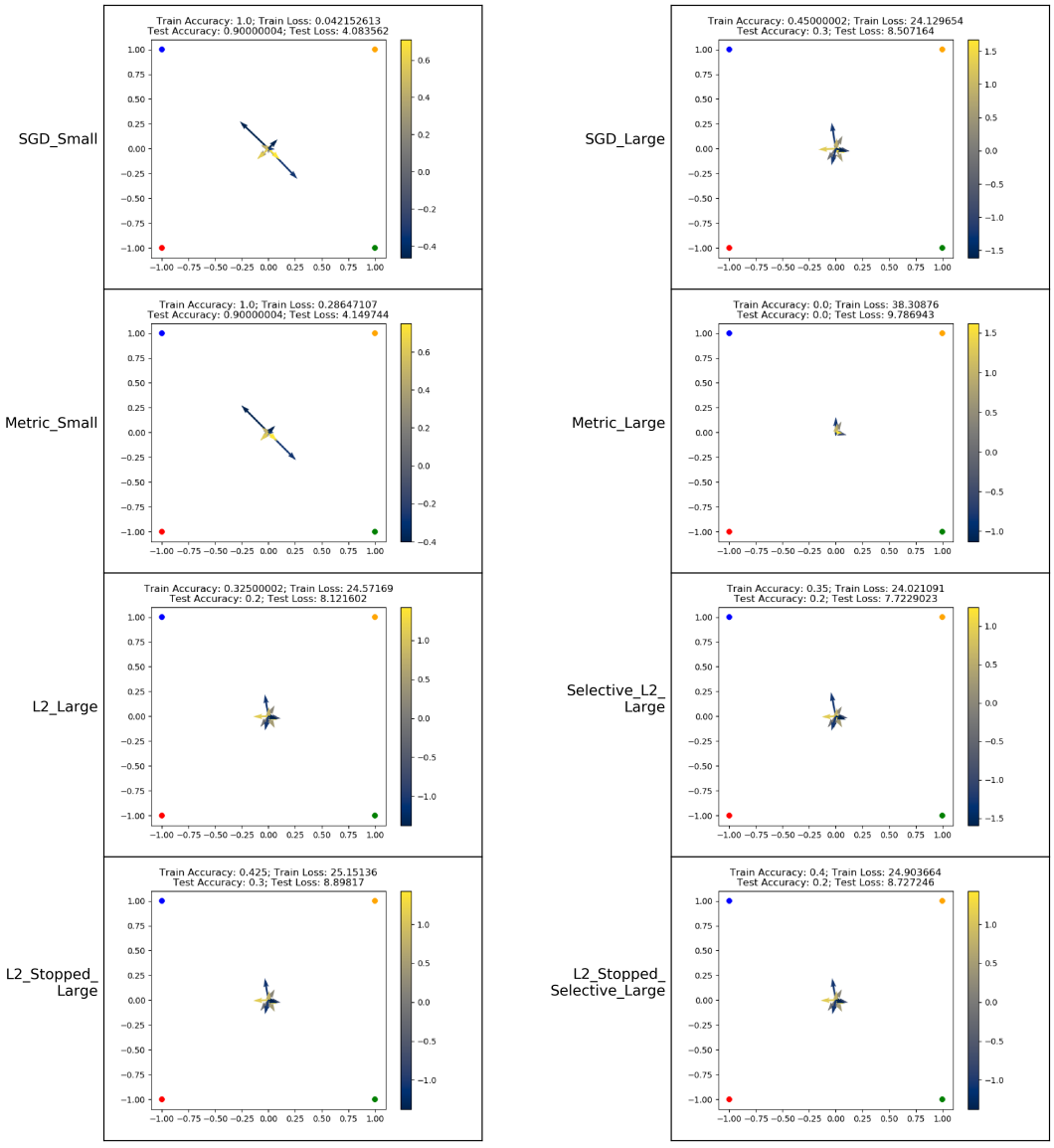


Figure 5.16: Plot of all input-hidden layer parameters from XORD data with ReLU Activation.

For all learning rules except the *Metric* learning rule we see that the ReLU activation does significantly better than the Tanh activation for this dataset. Tanh activation models learn faster from the small regime while the ReLU activation learns faster from the large regime. The discrepancy here in training speeds is likely due to the fact that ReLU does not saturate when pre-activations are large, while Tanh does. With both activations the speed of training is also the primary difference between the small and large regimes since the initialization regime has little effect on the final accuracies even between the *SGD* models. A final observation is that if *L2* is run for too long the training accuracy of the model begins to decrease. This effect is mitigated when using *Selective L2* (with a larger regularization rate when using Tanh, shown in Table 4.2) with Tanh and completely removed when using ReLU.

Looking at the test accuracies of the models for Tanh (Figure 5.18) and ReLU (Figure 5.21) we see that the effect of *L2* eventually decreasing the accuracy is also present but far less severe. In contrast using the *L2 Stopped* and *L2 Stopped Selective* learning rules again removes this effect entirely and both learning rules achieve superior training and test performance. The generalizability of *Selective L2* for MNIST is more evident with Tanh than ReLU activations. With Tanh it falls short of the small regime test accuracies by around a percent. With the ReLU activation, however, *Selective L2* does worse than all other regularizers except for *Metric* regularization but *L2 Stopped Selective* is still the best large regime model (tying with *Selective L2* when using Tanh). From these results it appears that regularizing all parameters in the early stages of training is beneficial, but it is necessary to switch to selective regularization to avoid over-regularizing. It is necessary to note the poor performance of the *Metric* regularizer irrespective of the activation function used. It appears that regularizing the parameters that are important to fitting the training data is detrimental to both training and test accuracy. Significantly, there is a difference in test performance between the small and large regimes. It is clear that the small regime always achieves superior test accuracy to the large regime when using *SGD*. This highlights the implicit regularization training from the small regime has on a model.

Two new observations can be made from the plots of the weight norms in Figures 5.19 and 5.22. Firstly we see that the epoch where *L2* regularization begins to have adverse effects is also near the epoch where the norm of the *L2* models intersects the norm of the vanilla *SGD* model from the small regime. This is an indication that training the model from the small regime naturally results in model parameters with close to the minimum possible norm. Secondly we note that in the Tanh plot of Figure 5.19 the models trained using the *L2*, *Selective L2* and *L2 Stopped Selective* learning rules from the large regime all have similar final weight norms. None of these models beat *SGD* from the small regime, supporting the previous point, but they also generalize differently between themselves. We see that while *Selective L2* and *L2 Stopped Selective* reduce the weight norm to a similar level as *L2*, the regularization to such a low norm is only detrimental to *L2*. This is further evidence that the negative effect of *L2* on training accuracy is due to the un-selective nature of the regularizer,

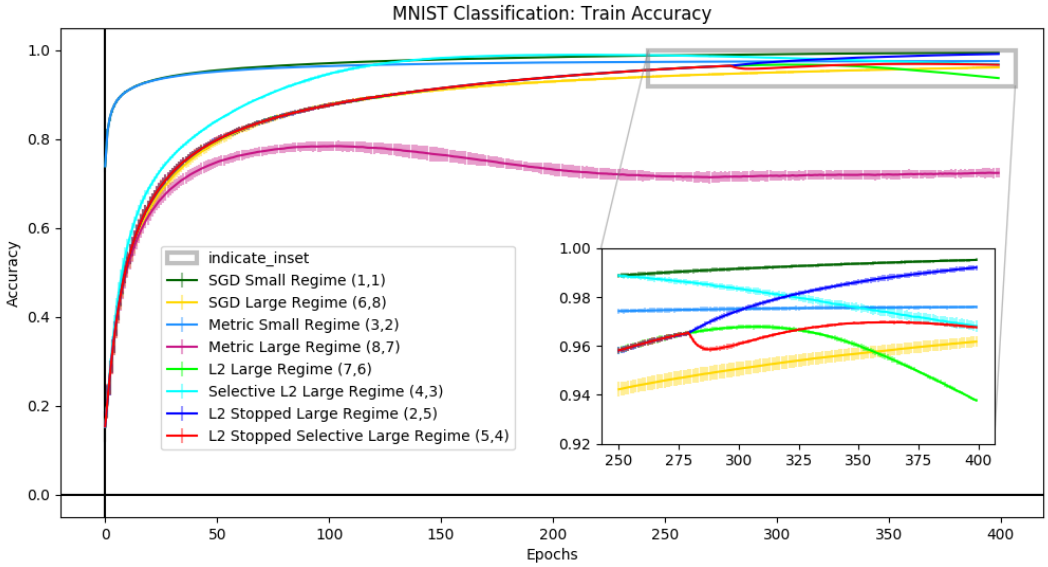


Figure 5.17: Mean Train Accuracy for MNIST with Tanh Activation over 10 runs. Shaded region is 2 standard deviations. Tuples in the legend represent the ranking of the learning rule based on maximum accuracy as (training data ranking, test data ranking).

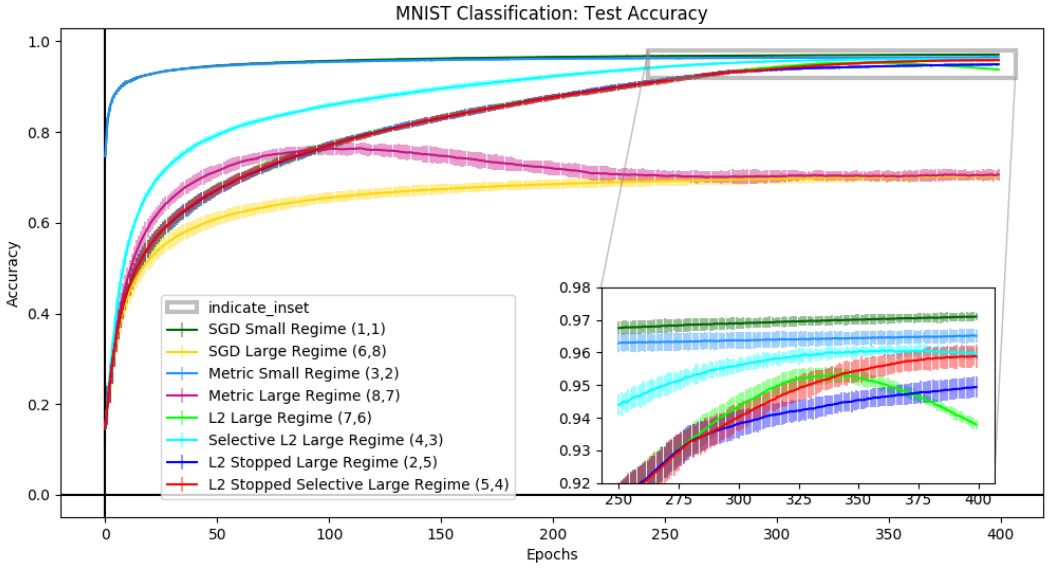


Figure 5.18: Mean Test Accuracy for MNIST with Tanh Activation over 10 runs. Shaded region is 2 standard deviations. Tuples in the legend represent the ranking of the learning rule based on maximum accuracy as (training data ranking, test data ranking).

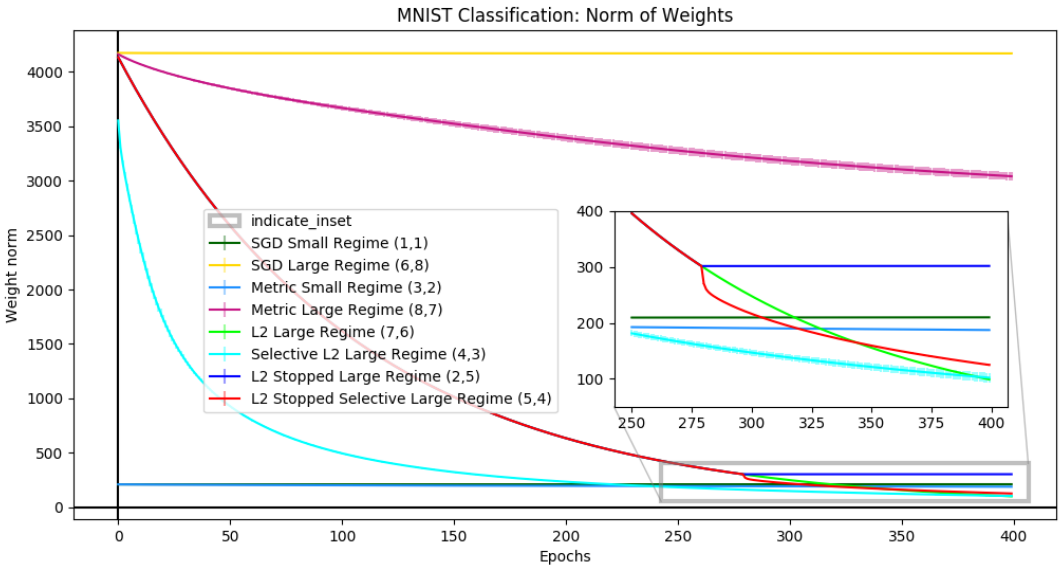


Figure 5.19: Mean Parameter Norms for MNIST with Tanh Activation over 10 runs. Shaded region is 2 standard deviations. Tuples in the legend represent the ranking of the learning rule based on maximum accuracy as (training data ranking, test data ranking).

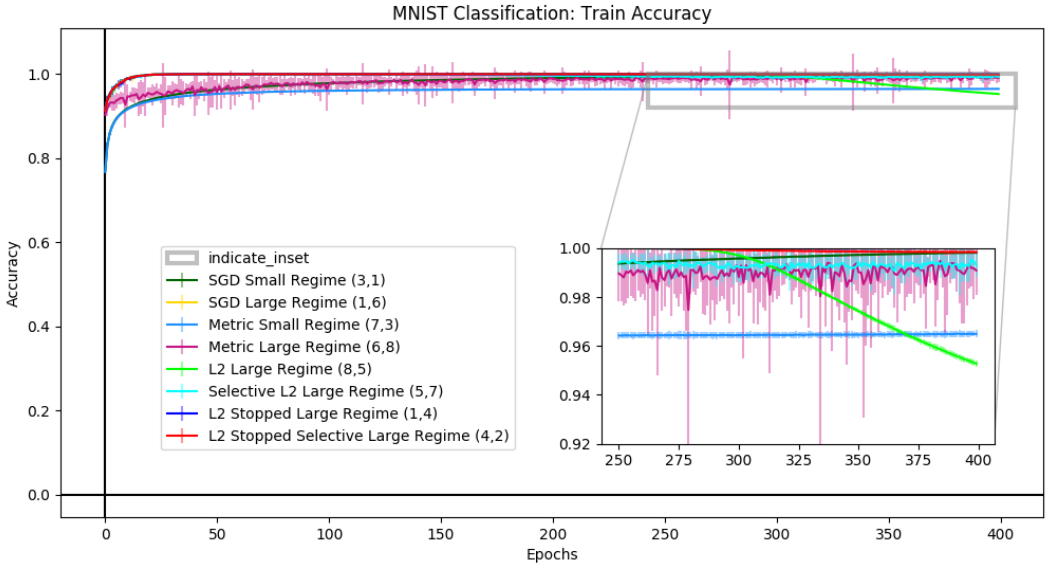


Figure 5.20: Mean Train Accuracy for MNIST with ReLU Activation over 10 runs. Shaded region is 2 standard deviations. Tuples in the legend represent the ranking of the learning rule based on maximum accuracy as (training data ranking, test data ranking).

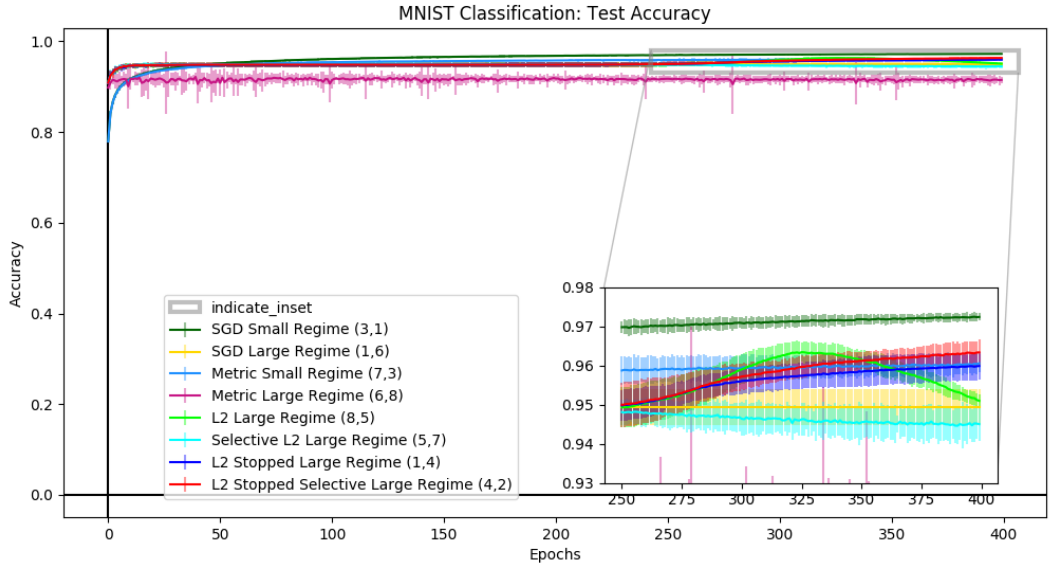


Figure 5.21: Mean Test Accuracy for MNIST with ReLU Activation over 10 runs. Shaded region is 2 standard deviations. Tuples in the legend represent the ranking of the learning rule based on maximum accuracy as (training data ranking, test data ranking).

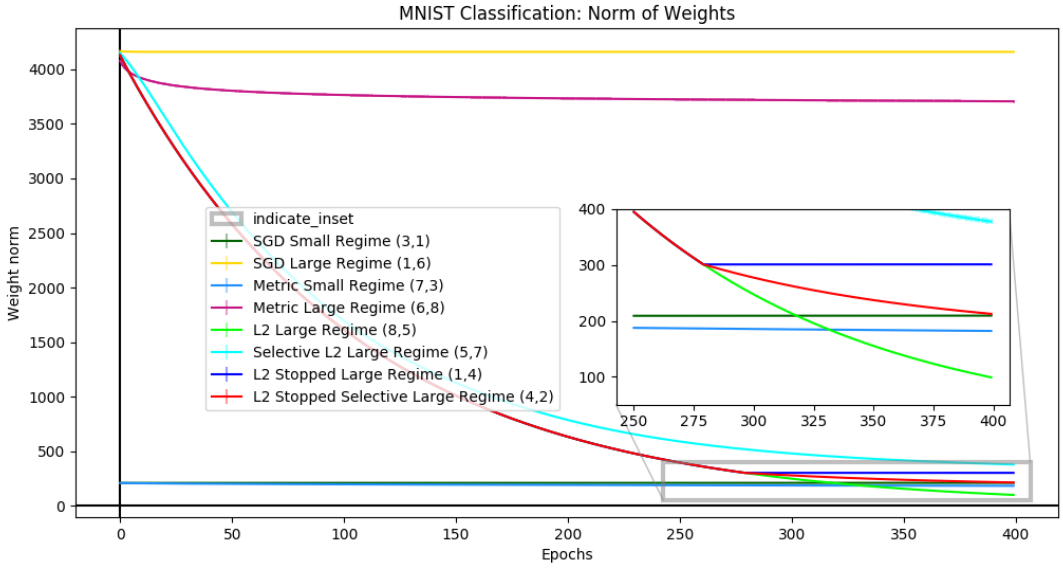


Figure 5.22: Mean Parameter Norms for MNIST with ReLU Activation over 10 runs. Shaded region is 2 standard deviations. Tuples in the legend represent the ranking of the learning rule based on maximum accuracy as (training data ranking, test data ranking).

and that by being selective we can remove the perceived trade-off between training and test data performance. The opposite effect is seen with *Metric* regularization on the large regime model. In this case we note that the final weight norm is significantly larger than the other regularized large regime models but we see a decrease in both training and test accuracy when using the *Metric* learning rule.

Ultimately the results on the MNIST dataset agree with those of the XOR and XORD results. The key points so far being, that small regime models generalize better than large regime models. Secondly, *Metric* regularization is damaging to training and test accuracy while some form of selective regularization can nearly replicate the generalizability of small regime models from a large initialization. Lastly, that while the norm of the model parameters is generally a good indication of generalizability, using fewer significant parameters to model the data is the effective way to reduce the norm compared to generally reducing all parameter values independently.

5.5 Synthetic Dataset

To discuss the results of the synthetic regression dataset we again begin by looking at the training error of both the Tanh and ReLU based models in Figures 5.23 and 5.26 respectively, since the observations are similar for both. Firstly, we see that *Metric* regularization has a detrimental effect on the training error regardless of the regime used to initialize the model. This effect is particularly clear in Figure 5.26 for the ReLU activation, where *L2* regularization also appears to increase training error. We do not observe a discrepancy in this case between the two initialization regimes aside from the fact that the large regime learns faster in the early stages of training. Looking at the norms for the ReLU models in Figure 5.28 we see the surprising result that while *L2 Stopped* and *L2 Stopped Selective* both get lower training errors, the norm of *L2 Stopped* is larger than *L2* (as expected since regularization is stopped) while *L2 Stopped Selective* continues to decrease the norm significantly. This result shows that it is not the fact that *L2* decreases the weight norm too severely for learning to take place, but rather that it does so indiscriminately which hinders the model from reaching lower training errors. This is supported by the fact that *Selective L2* is still able to achieve 0 training error with no variance by the end of training.

Looking at the test errors in Figures 5.24 and 5.27 we see the same observations noted for the other datasets. Namely that the large regime *SGD* model performs poorly and clearly overfits while small regime *SGD* obtains the lowest test error. We also see again that *Selective L2*, *L2 Stopped* and *L2 Stopped Selective* are capable of nearly matching the test error of small regime *SGD*. *L2* regularization also improves the generalizability of the models but not to the same degree as the other regularization methods mentioned, aside from metric regularization. This dataset quite clearly shows the source of the perception that there is a trade-off between training and test error. It appears from these results using

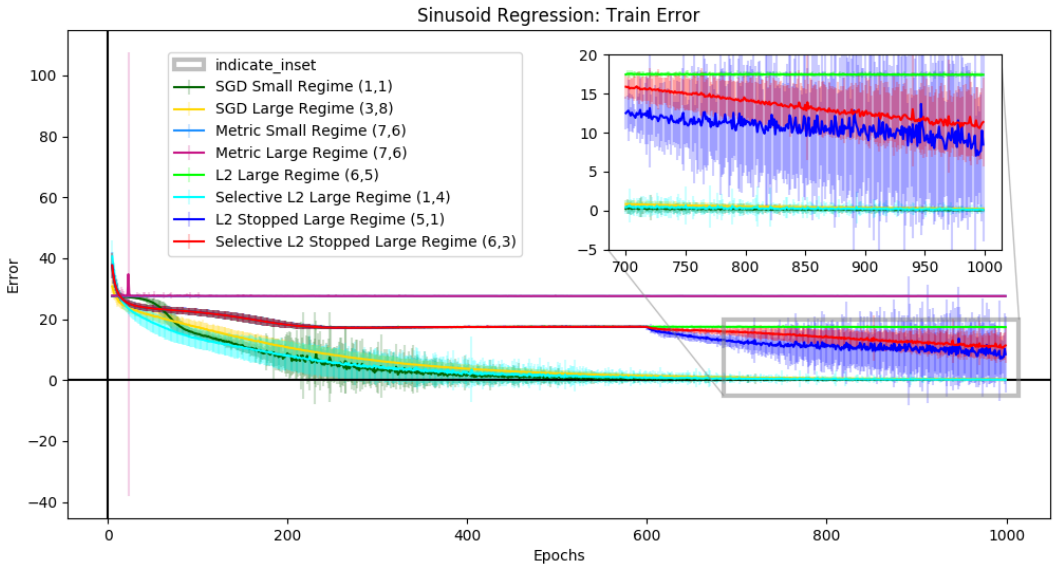


Figure 5.23: Mean Train Errors for Synthetic data with Tanh Activation over 30 runs. Shaded region is 2 standard deviations. Tuples in the legend represent the ranking of the learning rule based on minimum error as (training data ranking, test data ranking).

L2 regularization that this trade-off exists, however, we see with the other regularizers that it is possible to decrease the models' weight norms far more than what *L2* is able to, also decreasing test error significantly, without increasing training error. This also reflects the selective nature of the implicit regularization from small weight initialization, since on this dataset it appears that to obtain low training error, test error and weight norms the model would have to be selective about the number of significant parameters it uses, reinforcing our qualitative observations of the input-to-hidden layer weights on the XOR and XORD tasks.

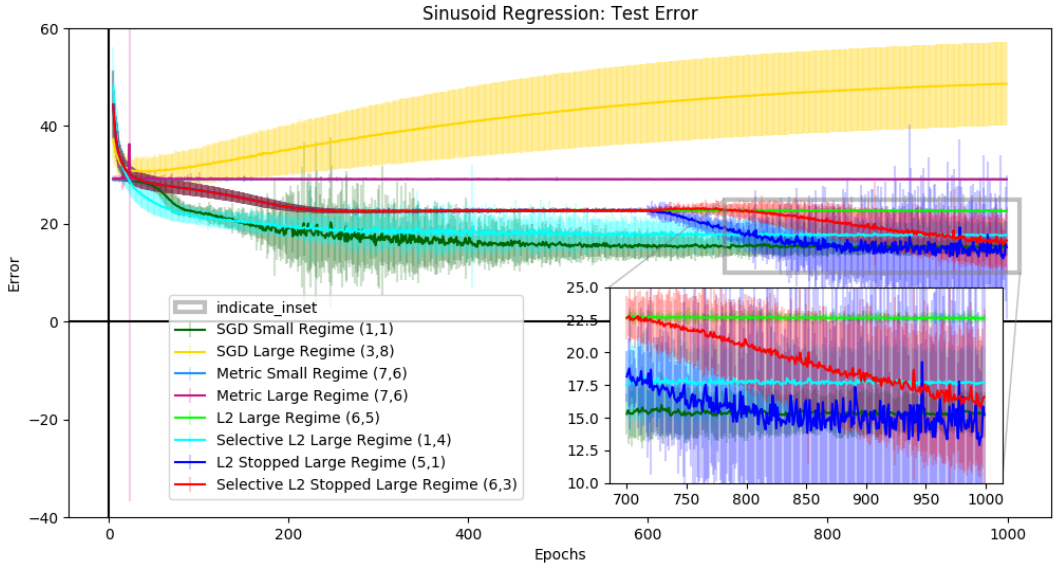


Figure 5.24: Mean Test Errors for Synthetic data with Tanh Activation over 30 runs. Shaded region is 2 standard deviations. Tuples in the legend represent the ranking of the learning rule based on minimum error as (training data ranking, test data ranking).

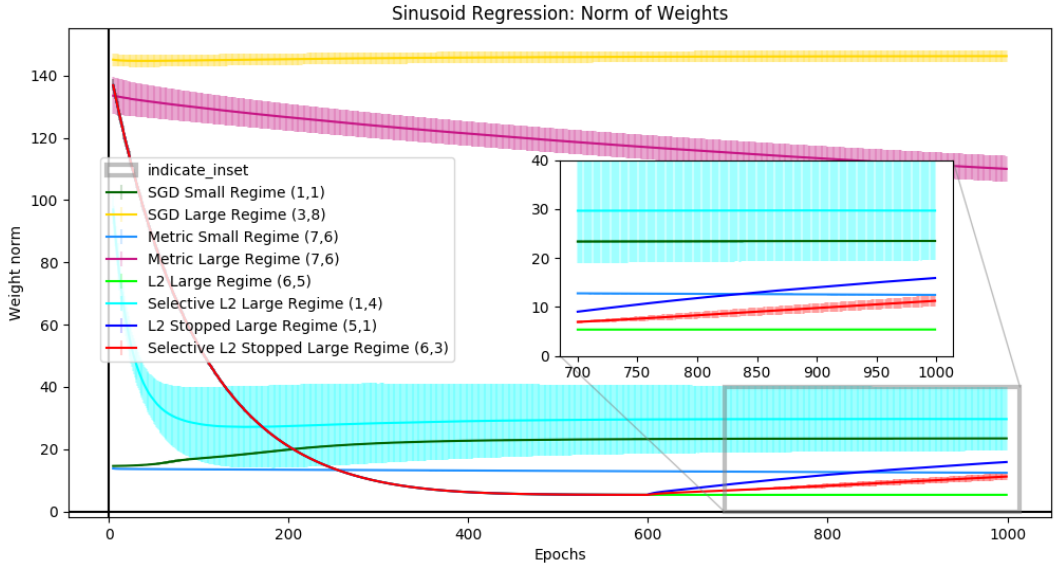


Figure 5.25: Mean Parameter Norms for Synthetic data with Tanh Activation over 30 runs. Shaded region is 2 standard deviations. Tuples in the legend represent the ranking of the learning rule based on minimum error as (training data ranking, test data ranking).

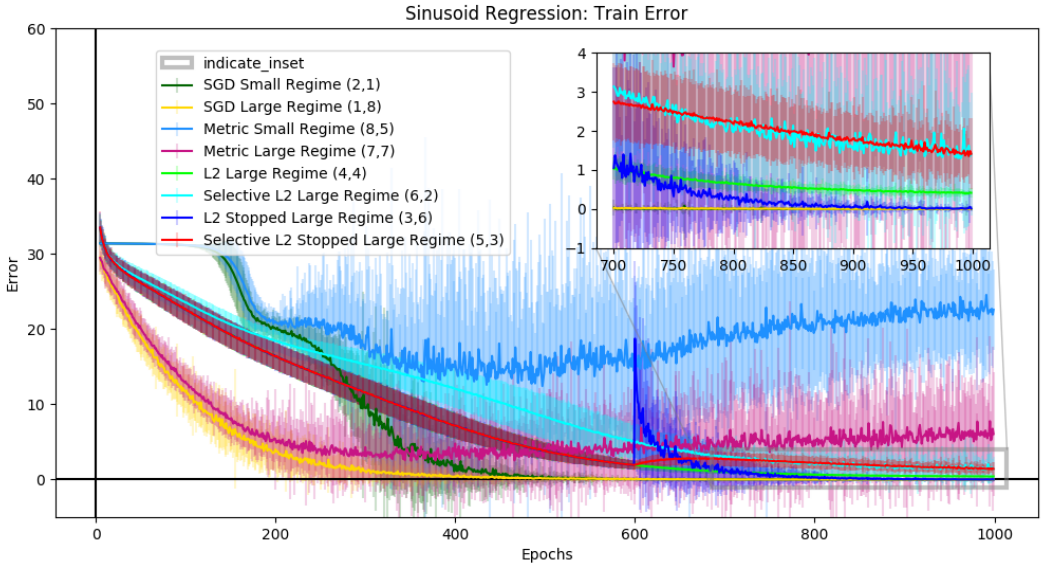


Figure 5.26: Mean Train Errors for Synthetic data with ReLU Activation over 30 runs. Shaded region is 2 standard deviations. Tuples in the legend represent the ranking of the learning rule based on minimum error as (training data ranking, test data ranking).

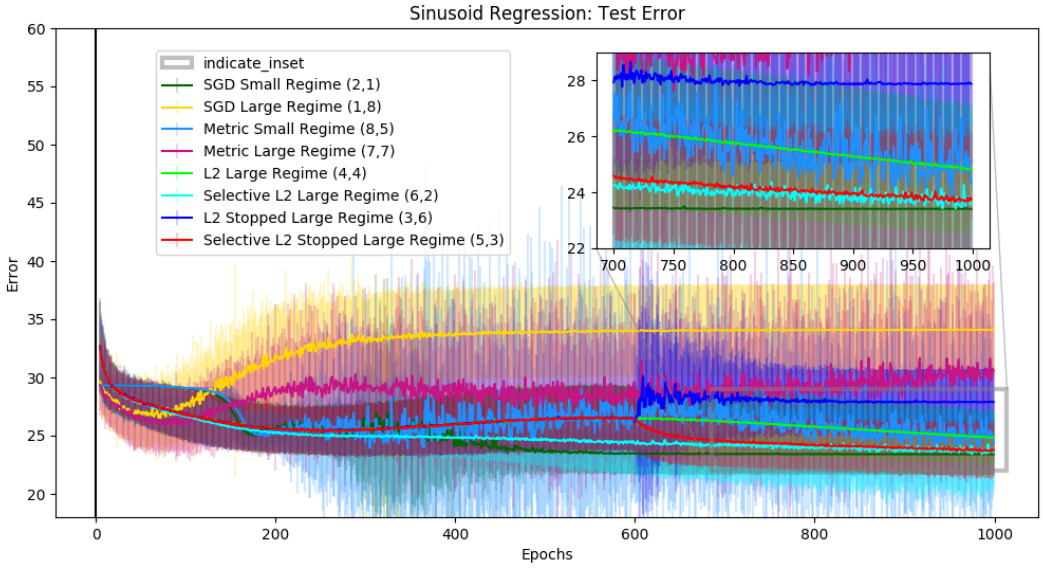


Figure 5.27: Mean Test Errors for Synthetic data with ReLU Activation over 30 runs. Shaded region is 2 standard deviations. Tuples in the legend represent the ranking of the learning rule based on minimum error as (training data ranking, test data ranking).

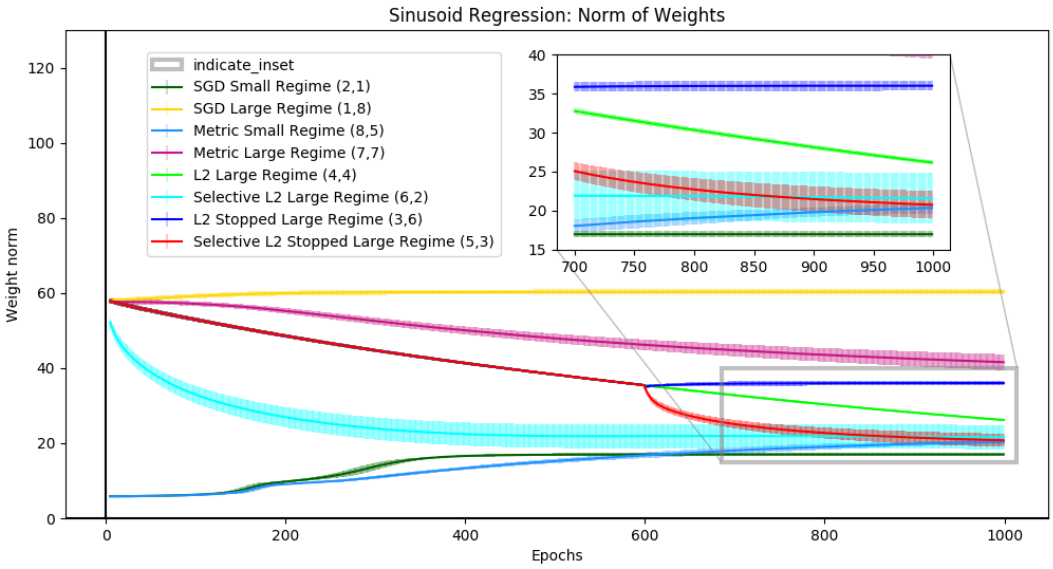


Figure 5.28: Mean Parameter Norms for Synthetic data with ReLU Activation over 30 runs. Shaded region is 2 standard deviations. Tuples in the legend represent the ranking of the learning rule based on minimum error as (training data ranking, test data ranking).

5.6 CIFAR-10 Dataset

We now present the results of our final experiment. We again begin with the training errors for the Tanh and ReLU models in Figures 5.29 and 5.32. We see that all learning rules are able to achieve 100% training accuracy on CIFAR-10 except for *SGD* from the large regime with ReLU activation. This is the result of a single bad run where the training diverged, which also explains the high variance of this learning rule. It appears that all forms of regularization increase the variance of individual epochs' accuracies. This results in the mean accuracies being lower than 100%, but we do see that at some point all of these models achieve a mean accuracy of 100% and all remain above 90% for the majority of training. The only regularizer that does not seem to increase training accuracy variance is the *Metric* regularizer. Looking at the norms in Figures 5.31 and 5.34 we see that the *Metric* regularizer had very little effect on the norm of the large regime model which indicates that when we set the hyper-parameters, increasing the *Metric* regularization rate made training unstable or resulted in the models diverging. For this dataset the optimal hyper-parameter resulted in a marginal impact of the *Metric* regularization compared to unregularized *SGD*. Finally, we see that the large regime initialization does speed up learning in the early stages of training compared to the small initialization regime.

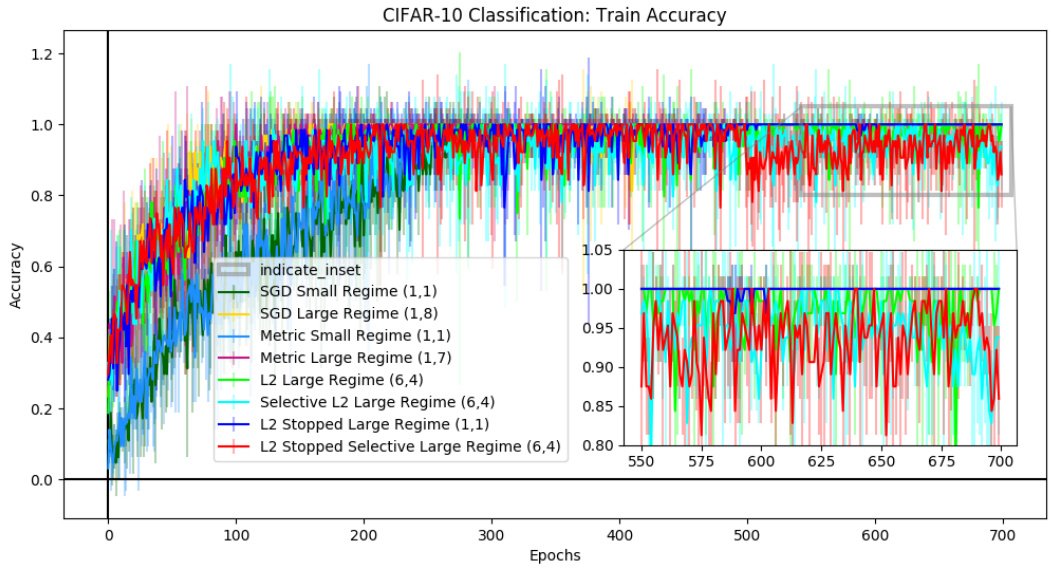


Figure 5.29: Mean Train Accuracy for CIFAR-10 with Tanh Activation over 10 runs. Shaded region is 2 standard deviations. Tuples in the legend represent the ranking of the learning rule based on maximum accuracy as (training data ranking, test data ranking).

Looking at the test accuracies in Figure 5.30 and 5.33 we see again that *SGD* and *Metric* regularization with large regime initialization generalize worse than all other learning rules. Aside from *Metric* regularization all regularization methods improve generalization. With Tanh activation these learning rules achieve similar test accuracy to the small regime *SGD* model, while with ReLU they are consistently 2% to 5% lower than the small regime *SGD* model. We do see, however, that *L2* regularization has higher variance in certain epochs than the other successful regularizers which indicates the fragility of the method. The fact that the successful regularizers generalize to a similar degree is likely due to the fact that the network used is quite small relative to the complexity of the dataset. Thus, the full capacity of the model was being used regardless of the regularizer, and controlling the magnitude of the parameter norm was all that was necessary. This model is still sufficient for us to observe the negative effect of large regime initialization on generalization and the inability of the *Metric* regularizer to offer a stable regularization method. A final observation from the norms of the parameters in Figure 5.31 is that *L2*, *L2 Stopped*, *Selective L2* and *L2 Stopped Selective* converge to lower norm solutions that small regime *SGD* while achieving comparable performance on both training and test accuracy. This indicates that in some cases small regime *SGD* does not always find the minimum-norm solution, motivating the use of regularizers, but also shows that within a certain range increasing the weight norm has little effect on the generalizability of the models.

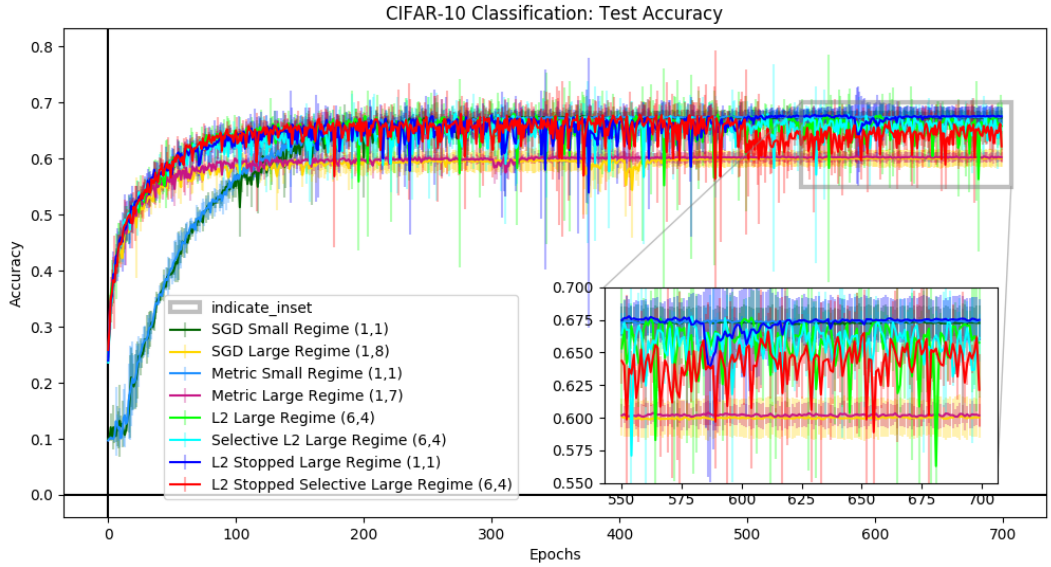


Figure 5.30: Mean Test Accuracy for CIFAR-10 with Tanh Activation over 10 runs. Shaded region is 2 standard deviations. Tuples in the legend represent the ranking of the learning rule based on maximum accuracy as (training data ranking, test data ranking).

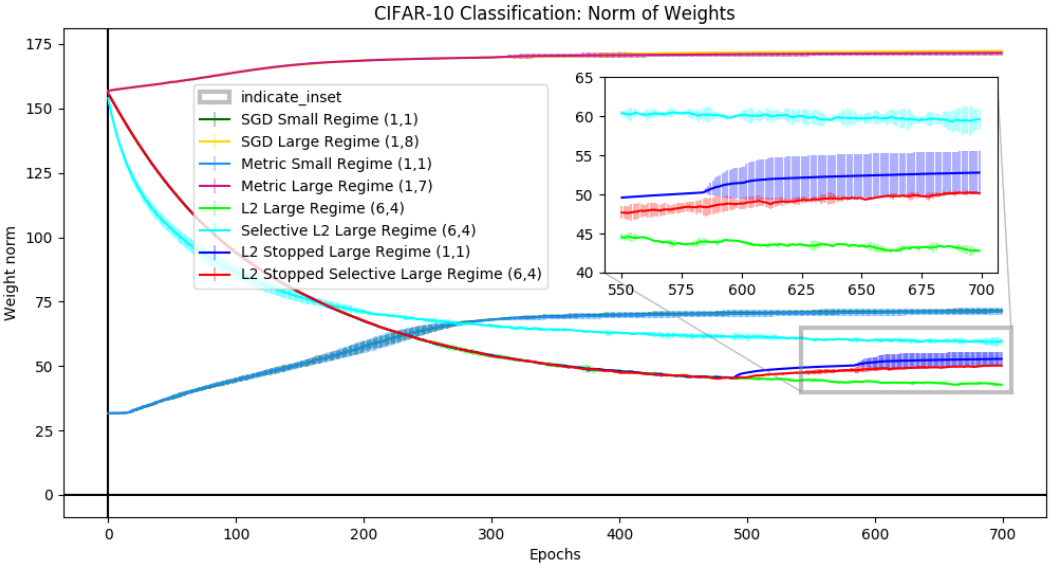


Figure 5.31: Mean Parameter Norms for CIFAR-10 with Tanh Activation over 10 runs. Shaded region is 2 standard deviations. Tuples in the legend represent the ranking of the learning rule based on maximum accuracy as (training data ranking, test data ranking).

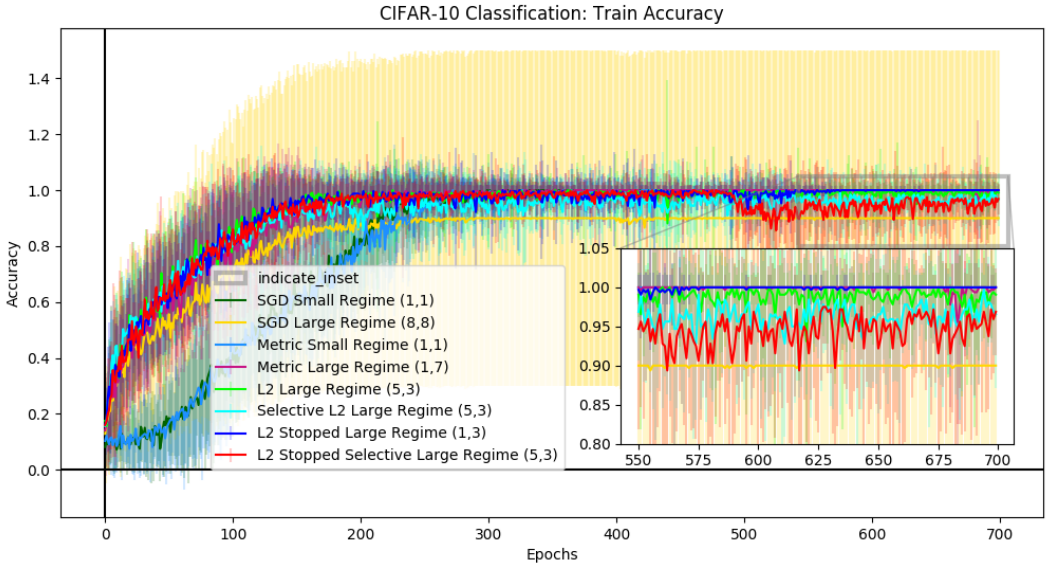


Figure 5.32: Mean Train Accuracy for CIFAR-10 with ReLU Activation over 10 runs. Shaded region is 2 standard deviations. Tuples in the legend represent the ranking of the learning rule based on maximum accuracy as (training data ranking, test data ranking).

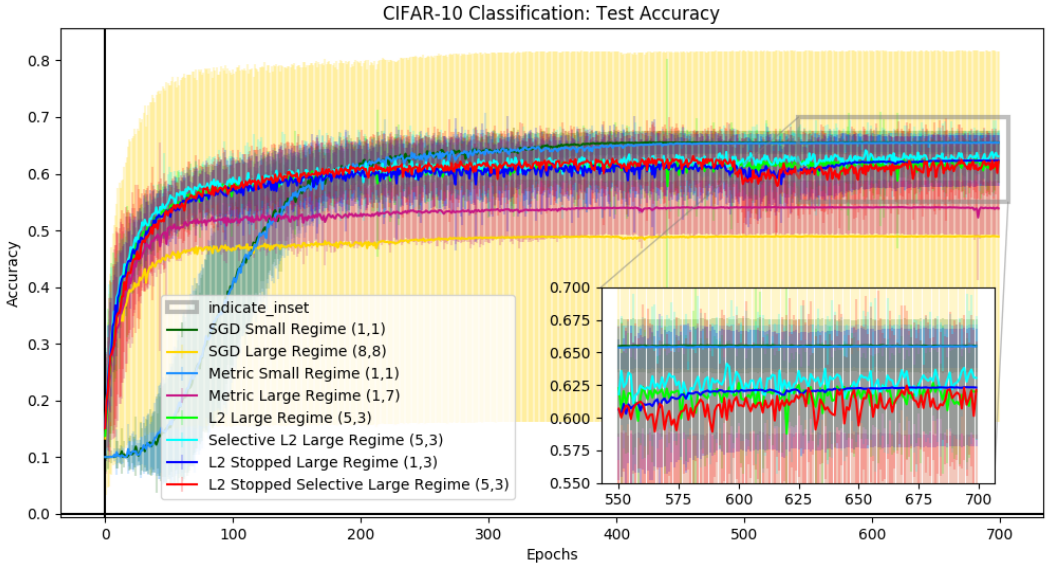


Figure 5.33: Mean Test Accuracy for CIFAR-10 with ReLU Activation over 10 runs. Shaded region is 2 standard deviations. Tuples in the legend represent the ranking of the learning rule based on maximum accuracy as (training data ranking, test data ranking).

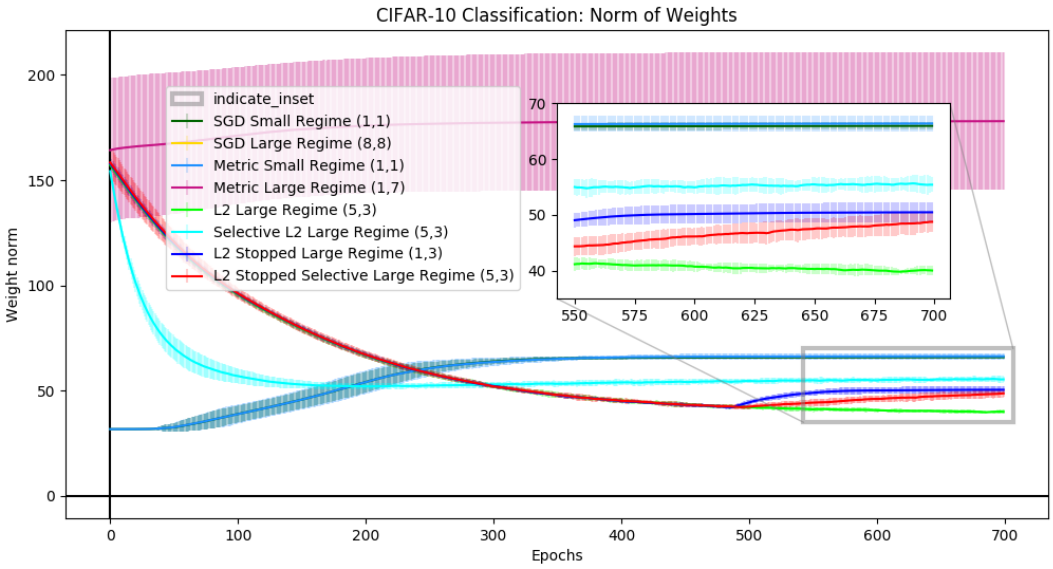


Figure 5.34: Mean Parameter Norms for CIFAR-10 with ReLU Activation over 10 runs. Shaded region is 2 standard deviations. Tuples in the legend represent the ranking of the learning rule based on maximum accuracy as (training data ranking, test data ranking).

5.7 Eigenvalues from Hessian of Loss

In this section we use the Memory Efficient Power Method shown in Algorithm 1 to calculate the Eigenvalues of the Hessian of the loss of our NNs. We present a histogram of the mean Eigenvalues in descending order with 2 standard deviations on the error bars. Eigenvalues are calculated at the end of training and are only calculated relative to the loss portion of any of the update rules, to avoid regularization terms from skewing the results. These Eigenvalues are averaged over the repeated trainings of the NNs as described in Section 4.4. For brevity we include a selection of these plots which are representative of our findings.

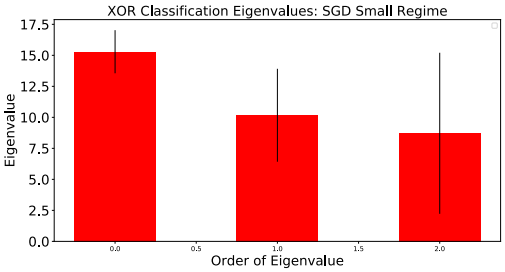
We first consider the results of the NN with the Tanh activation trained to perform the XOR task. These results are shown in Figure 5.35. Our first observation is that models trained from the small regime have lower Eigenvalues than large regime models. Since the other sections of this chapter showed the implicit regularization of small regime models and their natural propensity to generalize, this observation supports the notion that small Eigenvalues correspond to better generalizability of the model. This is also supported by the fact that the *SGD* and *Metric* learning rules from the large regime tend to generalize the worst and have the largest Eigenvalues in Figure 5.35. The *L2* and *Selective L2* learning rules from the large regime also have large Eigenvalues compared to the small regime models. These two learning rules, however, generalize well in our other experiments and *Selective L2* in particular often

matches the performance of small regime models on training and test data. This contradicts the notion that smaller Eigenvalues correspond to models which generalize better. The fact that *L2* and *Selective L2* have similar Eigenvalues also reflects that on this dataset the Eigenvalues do not reflect the difference between sparsity and low-norm methods of regularization. From these results it seems that small Eigenvalues are a strong predictor of a model which generalizes well, however, models with large Eigenvalues do not necessarily generalize poorly.

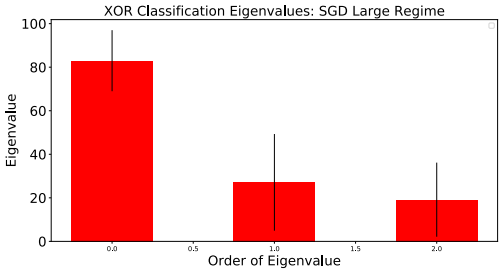
Now consider the results of the MNIST dataset with the Tanh activation shown in Figure 5.36. We again see that the small regime has smaller Eigenvalues, however, in this case the discrepancy between the regimes is far less severe. We see instead that the *Metric* and *Selective L2* learning rules with the large regime have far larger Eigenvalues than the other learning rules, even compared to *L2* regularization. These two learning rules have very different test accuracies on MNIST with the Tanh activation. As shown in Figure 5.18 *Selective L2* generalizes well and *Metric* regularization generalizes poorly. This is another result showing that large Eigenvalues are not necessarily correlated with poor generalizability. The difference in the Eigenvalues for *L2* and *Selective L2* in this case is also interesting as it seems that by being selective with the regularization we guide the model into sharper (less flat) areas of the loss landscape. This is likely due to *Selective L2* forcing the model to rely heavily on a subset of parameters and so the loss is particularly sensitive to these parameters. This then increases the curvature in the directions of parameter space corresponding to those parameters.

For the results on the Synthetic dataset with the ReLU activation shown in Figure 5.37, firstly we see that there is only one dominant Eigenvalue for each of the learning rules. This is in line with previous work [Gur-Ari *et al.* 2018] which showed that the number of significant Eigenvalues from the Hessian of the loss of an NN matches the number of output neurons of the NN. Secondly, for this dataset and architecture we see no discrepancy between the Eigenvalues from the small and large regimes with *SGD*. The two regimes are apparent when using *Metric* regularization. We also note that *Metric* regularization increases the variance in the Eigenvalues compared to *SGD*. This is significant as it shows the inconsistency in the final parametrization of the models trained with *Metric* regularization. These results again show that generalizability is not correlated to the Eigenvalues of the Hessian as *SGD* from the small regime generalizes significantly better than *SGD* from the large regime for this dataset, as can be seen in Figure 5.27.

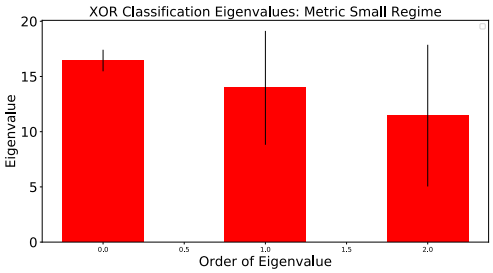
Finally, Figure 5.38 shows the results from the CIFAR-10 dataset with the NN using the ReLU activation function. These results support the observations above. It is apparent from these results that this dataset results in a large variance in the Eigenvalues regardless of the learning rules. The variance is particularly large with *Metric* regularization from the large regime. We see that this model achieves the tied-best training accuracy but second worst test accuracy. Thus, the variance in the Eigenvalues is likely due to the sensitivity of the



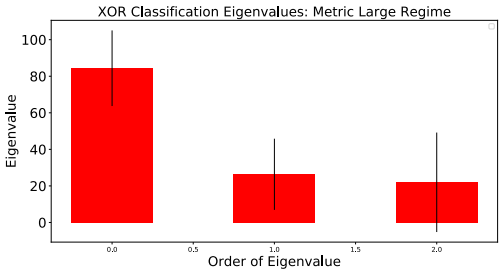
(a) *SGD* (Small Regime).



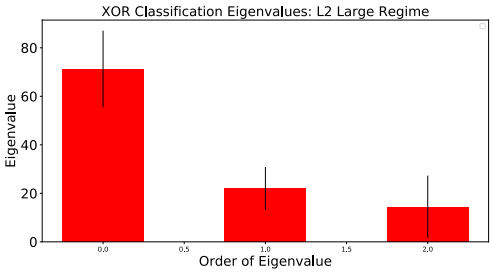
(b) *SGD* (Large Regime).



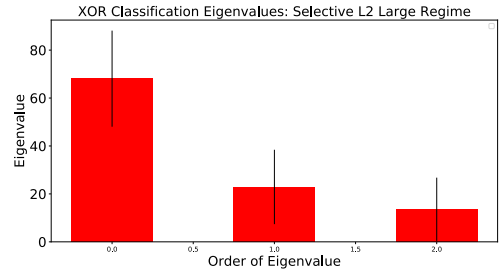
(c) *Metric* (Small Regime).



(d) *Metric* (Large Regime).

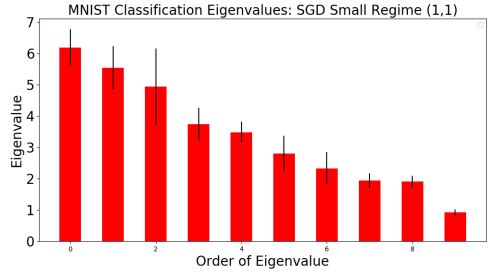


(e) L2 (Large Regime).

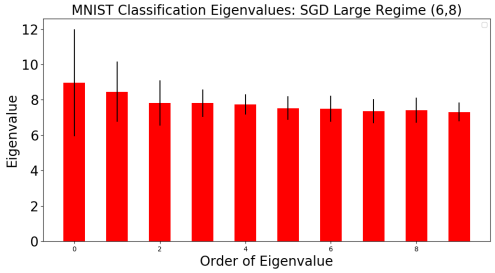


(f) *Selective L2* (Large Regime).

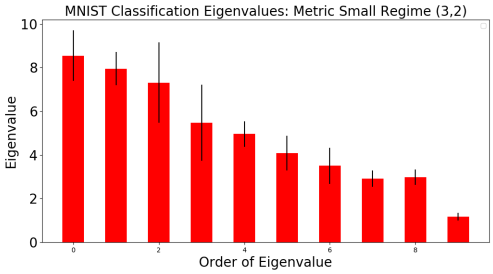
Figure 5.35: Means for the 3-top Eigenvalues from the Hessian of the loss function for the NN trained on the XOR task with the Tanh activation using the indicated learning rule. Black lines through the middle of each bar reflect 2 standard deviations. Note the difference in the scales of the y-axes of the plots.



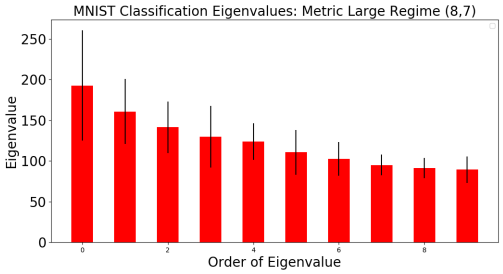
(a) *SGD* (Small Regime).



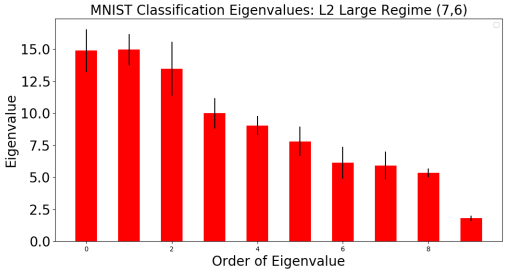
(b) *SGD* (Large Regime).



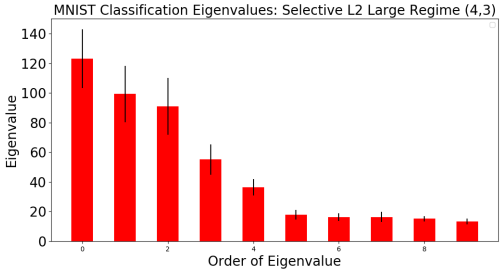
(c) *Metric* (Small Regime).



(d) *Metric* (Large Regime).

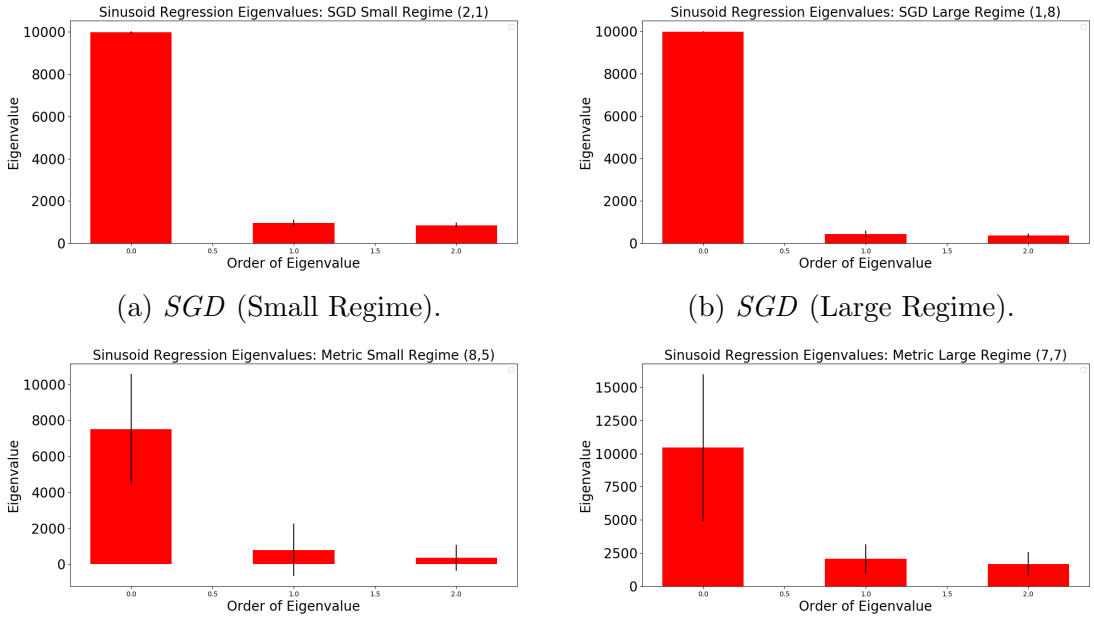


(e) L2 (Large Regime).



(f) *Selective L2* (Large Regime).

Figure 5.36: Means for the 10-top Eigenvalues from the Hessian of the loss function for the NN trained on MNIST with the Tanh activation using the indicated learning rule. Black lines through the middle of each bar reflect 2 standard deviations. Note the difference in the scales of the y-axes of the plots. Tuples in the title represent the ranking of the learning rule based on maximum accuracy in the original experiment as (training data ranking, test data ranking).



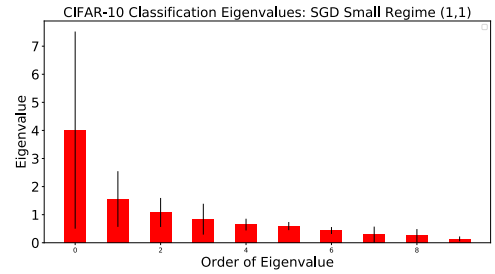
(a) *SGD* (Small Regime).

(b) *SGD* (Large Regime).

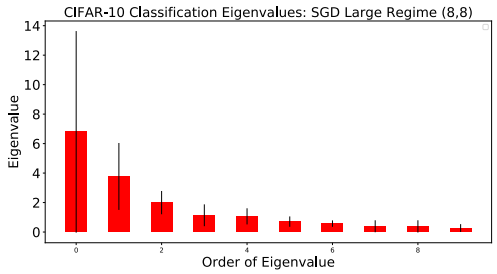
(c) *Metric* (Small Regime).

(d) *Metric* (Large Regime).

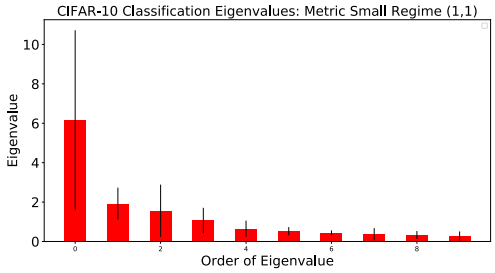
Figure 5.37: Means for the 10-top Eigenvalues from the Hessian of the loss function for the NN trained on the Synthetic dataset with the ReLU activation using the indicated learning rule. Black lines through the middle of each bar reflect 2 standard deviations. Note the difference in the scales of the y-axes of the plots. Tuples in the title represent the ranking of the learning rule based on minimum error in the original experiment as (training data ranking, test data ranking).



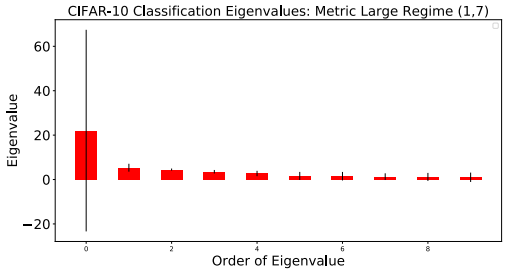
(a) *SGD* (Small Regime).



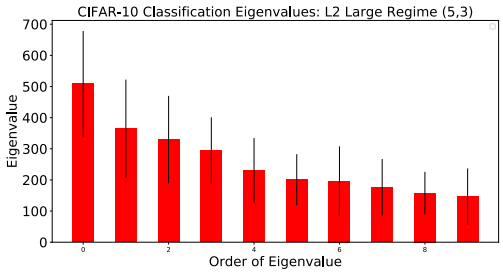
(b) *SGD* (Large Regime).



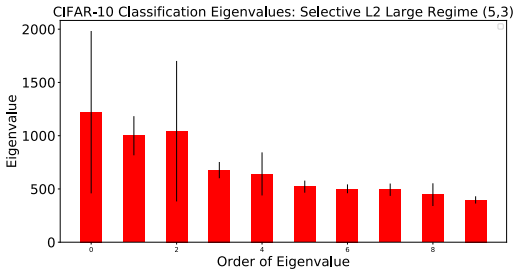
(c) *Metric* (Small Regime).



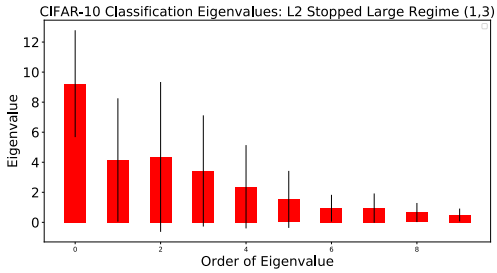
(d) *Metric* (Large Regime).



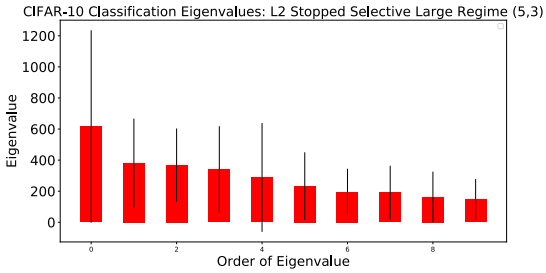
(e) L2 (Large Regime).



(f) *Selective L2* (Large Regime).



(g) L2 Stopped (Large Regime).



(h) L2 Stopped Selective (Large Regime).

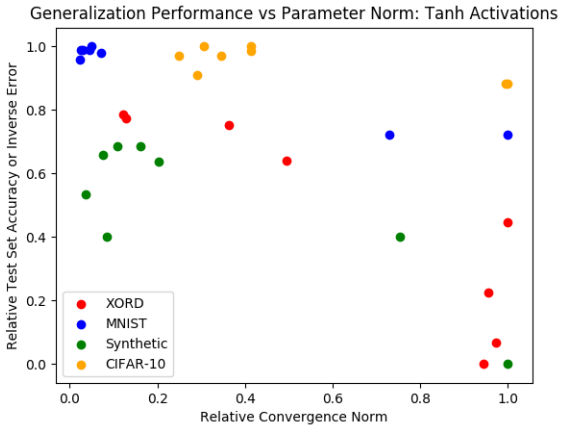
Figure 5.38: Means for the 10-top Eigenvalues from the Hessian of the loss function for the NN trained on CIFAR-10 with the ReLU activation using the indicated learning rule. Black lines through the middle of each bar reflect 2 standard deviations. Note the difference in the scales of the y-axes of the plots. Tuples in the title represent the ranking of the learning rule based on maximum accuracy as (training data ranking, test data ranking).

parametrizations found using *Metric* regularization to the training data and initialization. As a result the parameters learned by the model between runs are inconsistent and have very different final Eigenvalues for the Hessian. We also note that in spite of the larger and more complicated architecture we use for this dataset, the range of the Eigenvalues is consistent with the previous results. In particular the smaller Eigenvalues are similar to the Eigenvalues from the MNIST dataset. The larger Eigenvalues, however, are similar to the Synthetic dataset. Yet, both of the MNIST and Synthetic architectures are significantly smaller than the CIFAR-10 architectures. Thus, the Eigenvalues of the Hessian appears to be an architecture-size agnostic method for analysing an NN. This is a useful property as it allows us to compare the results of the different datasets in spite of different network architectures being used for each one. There is still an apparent difference in the range of Eigenvalues between the learning rules for CIFAR-10. This is likely due to the higher complexity of the CIFAR-10 dataset and as a result the complexity of the loss landscape. We again see that comparing the Eigenvalues in Figure 5.38 with the test data accuracies of the learning rules in Figure 5.33 that the Eigenvalues are not consistent predictors of the generalizability of the model. In particular we see that regularization tends to increase the Eigenvalues but improves the generalizability of the models. We also note that the *SGD* and *Metric* learning rules from both the small and large initialization regimes have significantly different test data accuracies but have a similar scale and distribution to their Eigenvalues.

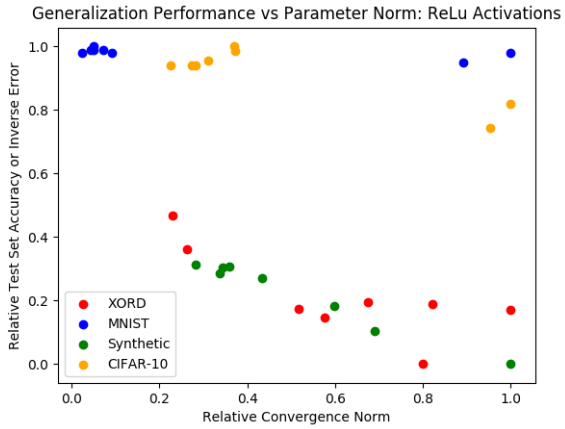
5.8 Conclusion

In conclusion we find the following observations from our empirical results. Firstly, *Metric* regularization is detrimental to both training and test performance irrespective of the dataset or architecture. The XOR and XORD results indicate that this is due to the fact that *Metric* regularization stops the model from learning to rely on a small set of significant features or patterns. Instead *Metric* regularization uses more parameters which capture fewer explicit features in the training dataset. This observation is supported by the generally good performance of *Selective L2* compared to *Metric* regularization. Thus, we conclude that adaptively regularizing the parameters which are more significant for reducing the training loss does not reduce the generalization gap of the NN and hinders the model from fitting the training data. Instead adaptively regularizing the insignificant parameters reduces the generalization gap with no detrimental effect on the training loss.

Secondly, by comparing *L2* with *Selective L2* (and *L2 Stopped* with *L2 Stopped Selective*) we see that the perceived trade-off between training and test data performance is not necessary and that the detrimental effect of *L2* regularization on training data performance is due to the fact that it (similar to *Metric* regularization) regularizes parameters which are significant for fitting the training data. We see that this does not improve the generalizability of the model but decreases the training data performance. *L2* does still regularize the insignificant



(a) Generalization vs Parameter-Norm when using Tanh Activation



(b) Generalization vs Parameter-Norm when using ReLU Activation

Figure 5.39: Plots reflecting the correlation between lower parameter-norms and improved test data performance. For each learning rule we plot the final relative parameter norm vs the final relative test data performance. For classification tasks the normalized accuracy is used for test data performance, while regression tasks we multiply the final test error (MSE) by -1, normalize and then add 1. This is done to make higher test data performance better for the regression tasks on this axis. To normalize we divide all values by the max value in the set.

parameters which improves the generalizability of the model. These two separate effects being observed from *L2* gives the appearance of a trade-off, but we can remove all negative effects on the training data performance by being selective about regularization without hindering the test data performance.

Lastly, as shown in Figure 5.39, we see that the notion of low-norm parametrizations generalizing better than higher-norm parametrizations is accurate. This is shown by the trend of the scatter plots to move from the top-left to bottom-right of the plane. The parameter-norm vs generalizability relationship is most prominent for regression tasks compared to classification tasks. This effect, however, is limited in two ways. Firstly, past a certain point lowering the parameter norm does not aid generalizability and can begin to hinder the model. There is a range of good parameter norms with little difference in the generalizability of models within this range. Secondly, based on the individual results in previous sections, we see that it is better for generalizability to reduce the parameter norm by reducing insignificant parameters more severely and leaving a few significant parameters unaffected. This is in contrast to reducing the parameter norm by decreasing all model parameters. Thus, while the parameter norm is correlated with better model generalizability, it does not accurately reflect the entire dynamics of the model and is an incomplete measure of the generalizability of the model.

The inability of the parameter norm to fully predict generalizability is apparent in the discrepancy in the generalizability of models which start near the low-norm model parameters and models which are regularized into the same low-norm region. Our results agree with previous work [Geiger *et al.* 2020] that shows that the large regime models generalize worse than small regime models. An unintuitive result we observe is that *L2* regularization is unable to reproduce the performance of the small regime models on test data. Based on the results from the XOR and XORD datasets, as well as the *Selective L2* regularizer on all datasets, we can see that this is due to the fact that the small regime models use far fewer significant parameters than large regime models even when regularized with *L2*. Only *Selective L2* based regularizers were able to recover near small regime performance for all datasets even when trained from the large regime. This provides a valuable insight into the natural ability of NNs to generalize as we see that not only do small regime models find low-norm solutions, they also naturally maintain sparsity in their parameters. This is a powerful implicit regularization strategy as the model will naturally only learn the parameters which are necessary for fitting the training data but avoid overfitting by not increasing its capacity more than necessary. A summary of all results for the networks using Tanh activation are shown in Table 5.1 and using ReLU activation in Table 5.2.

Table 5.1: Summary of final Training and Test Accuracy/Error (A/E in dataset name) and epoch time for each learning rule and dataset with the Tanh Network.

Learning Rule	Dataset	Training A/E	Test A/E	Param Norm	Epoch Time (s)
SGD Small	XOR (A)	1.0 ± 0	NA	7.18 ± 0.29	0.0001 ± 0.0061
	XORD (E)	0.74 ± 0.47	17.64 ± 4.14	12.18 ± 0.4	0.0001 ± 0.03
	MNIST (A)	0.995 ± 0.003	0.97 ± 0.0006	209.89 ± 0.12	0.31 ± 0.04
	Synthetic (E)	0.107 ± 0.107	15.22 ± 0.877	23.49 ± 0.14	0.148 ± 0.073
	CIFAR-10 (A)	1.0 ± 0.0	0.673 ± 0.0065	71.35 ± 0.44	0.8 ± 0.018
SGD Large	XOR (A)	1.0 ± 0	NA	27.08 ± 4.61	0.0001 ± 0.0062
	XORD (E)	0.0001 ± 0.0002	60.8 ± 51.3	90.88 ± 10.43	0.0001 ± 0.05
	MNIST (A)	0.96 ± 0.0011	0.703 ± 0.004	4170.87 ± 2.38	0.32 ± 0.03
	Synthetic (E)	0.252 ± 0.086	48.64 ± 4.26	146.22 ± 0.91	0.145 ± 0.07
	CIFAR-10 (A)	1.0 ± 0.0	0.6 ± 0.0008	172.3 ± 0.1	0.82 ± 0.01
Metric Small	XOR (A)	1.0 ± 0	NA	6.83 ± 0.27	0.0001 ± 0.004
	XORD (E)	1.675 ± 0.62	16.9 ± 3.39	11.59 ± 0.417	0.0004 ± 0.08
	MNIST (A)	0.976 ± 0.003	0.96 ± 0.0009	187.21 ± 0.29	0.4 ± 0.03
	Synthetic (E)	27.57 ± 0.0003	29.09 ± 0.0002	12.46 ± 0.1	0.15 ± 0.073
	CIFAR-10 (A)	1.0 ± 0.0	0.6751 ± 0.0	71.23 ± 6.89	1.19 ± 0.2
Metric Large	XOR (A)	1.0 ± 0.075	NA	24.12 ± 4.43	0.0001 ± 0.0038
	XORD (E)	0.0004 ± 47.67	43.36 ± 0.47	95.05 ± 14.5	0.01 ± 0.148
	MNIST (A)	0.724 ± 0.005	0.706 ± 0.006	3041.36 ± 18.67	0.4 ± 0.03
	Synthetic (E)	27.56 ± 0.002	29.09 ± 0.005	110.02 ± 2.1	0.148 ± 0.071
	CIFAR-10 (A)	1.0 ± 0.0	0.6 ± 0.004	171.53 ± 0.46	1.18 ± 0.01
L2	XOR (A)	1.0 ± 0	NA	18.44 ± 3.08	0.0001 ± 0.002
	XORD (E)	1.0 ± 0	73.3 ± 58.8	92.41 ± 15.48	0.00005 ± 0.025
	MNIST (A)	0.937 ± 0.0003	0.934 ± 0.0007	98.82 ± 0.05	0.32 ± 0.03
	Synthetic (E)	17.41 ± 0.09	22.67 ± 0.16	5.38 ± 0.01	0.148 ± 0.073
	CIFAR-10 (A)	1.0 ± 0.0	0.658 ± 0.02	42.79 ± 0.03	0.85 ± 0.06
Selective L2	XOR (A)	0.975 ± 0.075	NA	16.41 ± 5.06	0.0002 ± 0.0055
	XORD (E)	0.75 ± 0.23	19.31 ± 4.9	34.61 ± 4.13	0.0004 ± 0.07
	MNIST (A)	0.968 ± 0.0011	0.96 ± 0.0008	101.9 ± 4.36	0.45 ± 0.02
	Synthetic (E)	0.189 ± 0.186	17.68 ± 1.38	29.69 ± 5.09	0.151 ± 0.073
	CIFAR-10 (A)	0.9375 ± 0.0625	0.66 ± 0.009	59.59 ± 0.64	1.35 ± 0.25
L2 Stopped	XOR (A)	1.0 ± 0	NA	20.7 ± 3.48	0.0001 ± 0.0
	XORD (E)	0.0025 ± 0.0074	78.53 ± 82.5	89.72 ± 13.29	0.0002 ± 0.0
	MNIST (A)	0.992 ± 0.0005	0.949 ± 0.0015	301.89 ± 0.17	0.3 ± 0.001
	Synthetic (E)	8.46 ± 6.21	15.23 ± 7.33	15.91 ± 0.05	0.148 ± 0.073
	CIFAR-10 (A)	1.0 ± 0.0	0.675 ± 0.0078	52.79 ± 1.37	0.9 ± 0.02
L2 Stopped Selective	XOR (A)	1.0 ± 0	NA	15.23 ± 3.85	0.0001 ± 0.0
	XORD (E)	0.62 ± 0.48	28.33 ± 28.32	47.09 ± 7.75	0.0003 ± 0.0
	MNIST (A)	0.968 ± 0.0003	0.959 ± 0.0017	124.78 ± 0.72	0.37 ± 0.1
	Synthetic (E)	11.36 ± 2.82	16.59 ± 3.37	11.24 ± 0.57	0.149 ± 0.07
	CIFAR-10 (A)	0.859 ± 0.0469	0.62 ± 0.0013	50.13 ± 0.06	1.1 ± 0.2

Table 5.2: Summary of final Training and Test Accuracy/Error (A/E in dataset name) and epoch time for each learning rule and dataset with the ReLU Network.

Learning Rule	Dataset	Training A/E	Test A/E	Param Norm	Epoch Time (s)
SGD Small	XOR (A)	1.0 ± 0	NA	5.0 ± 0.18	0.0 ± 0.0061
	XORD (E)	0.0 ± 0	6.60 ± 1.86	8.35 ± 0.36	0.0 ± 0.05
	MNIST (A)	0.998 ± 0	0.97 ± 0.0004	209.39 ± 0.18	0.44 ± 0.05
	Synthetic (E)	0.0 ± 0.0	23.41 ± 0.88	16.99 ± 0.2	0.149 ± 0.05
	CIFAR-10 (A)	1.0 ± 0.0	0.66 ± 0.01	65.96 ± 0.34	0.8 ± 0.02
SGD Large	XOR (A)	1.0 ± 0	NA	24.49 ± 2.92	0.0 ± 0.0062
	XORD (E)	17.87 ± 1.96	8.56 ± 0.97	31.89 ± 3.24	0.0 ± 0.05
	MNIST (A)	1.0 ± 0	0.95 ± 0.002	4160.49 ± 3.65	0.42 ± 0.04
	Synthetic (E)	0.002 ± 0.003	34.09 ± 1.96	60.29 ± 0.33	0.148 ± 0.047
	CIFAR-10 (A)	0.9 ± 0.3	0.49 ± 0.16	169.67 ± 0.7	0.82 ± 0.01
Metric Small	XOR (A)	1.0 ± 0	NA	4.84 ± 0.21	0.0 ± 0.004
	XORD (E)	0.20 ± 0.16	5.50 ± 1.33	7.3 ± 0.19	0.01 ± 0.14
	MNIST (A)	0.96 ± 0	0.96 ± 0.001	181.96 ± 0.29	0.4 ± 0.01
	Synthetic (E)	22.58 ± 2.69	24.31 ± 1.79	20.36 ± 0.36	0.152 ± 0.05
	CIFAR-10 (A)	1.0 ± 0.0	0.65 ± 0.008	66.38 ± 0.7	1.19 ± 0.2
Metric Large	XOR (A)	0.95 ± 0.15	NA	22.16 ± 2.54	0.0 ± 0.0038
	XORD (E)	38.65 ± 3.84	10.32 ± 0.8	25.53 ± 2.49	0.01 ± 0.14
	MNIST (A)	0.99 ± 0.005	0.92 ± 0.004	3706.94 ± 6.79	0.4 ± 0.008
	Synthetic (E)	6.06 ± 4.13	30.58 ± 3.79	41.57 ± 1.06	0.152 ± 0.045
	CIFAR-10 (A)	1.0 ± 0.0	0.54 ± 0.02	178.02 ± 16.35	1.18 ± 0.01
L2	XOR (A)	1.0 ± 0	NA	16.77 ± 2.03	0.0 ± 0.0025
	XORD (E)	21.28 ± 1.63	8.30 ± 0.77	21.52 ± 2.18	0.0 ± 0.04
	MNIST (A)	0.95 ± 0	0.95 ± 0.0007	98.86 ± 0.08	0.34 ± 0.01
	Synthetic (E)	0.42 ± 0.06	24.83 ± 0.15	26.17 ± 0.13	0.149 ± 0.045
	CIFAR-10 (A)	0.99 ± 0.01	0.62 ± 0.02	40.03 ± 0.46	0.85 ± 0.06
Selective L2	XOR (A)	1.0 ± 0	NA	14.85 ± 3.84	0.0 ± 0.0055
	XORD (E)	22.30 ± 3.14	8.80 ± 1.25	18.39 ± 4.45	0.01 ± 0.14
	MNIST (A)	0.99 ± 0.002	0.95 ± 0.002	377.67 ± 2.54	0.48 ± 0.01
	Synthetic (E)	1.41 ± 0.79	23.60 ± 1.82	21.60 ± 1.62	0.151 ± 0.047
	CIFAR-10 (A)	0.97 ± 0.05	0.63 ± 0.02	55.41 ± 0.67	1.35 ± 0.25
L2 Stopped	XOR (A)	1.0 ± 0	NA	18.80 ± 2.27	0.0 ± 0.0
	XORD (E)	19.26 ± 1.84	8.37 ± 1.03	26.21 ± 2.66	0.0 ± 0.0
	MNIST (A)	1.0 ± 0	0.96 ± 0.002	301.13 ± 0.26	0.33 ± 0.01
	Synthetic (E)	0.02 ± 0.03	27.88 ± 1.43	36.0 ± 0.33	0.145 ± 0.073
	CIFAR-10 (A)	1.0 ± 0.0	0.62 ± 0.02	50.47 ± 0.88	0.85 ± 0.02
L2 Stopped Selective	XOR (A)	1.0 ± 0	NA	12.63 ± 2.55	0.0 ± 0.0
	XORD (E)	22.79 ± 2.04	8.54 ± 0.71	16.49 ± 3.19	0.0 ± 0.0
	MNIST (A)	1.0 ± 0	0.96 ± 0.002	212.54 ± 0.75	0.37 ± 0.1
	Synthetic (E)	1.42 ± 0.46	23.75 ± 1.16	20.75 ± 0.88	0.15 ± 0.046
	CIFAR-10 (A)	0.97 ± 0.03	0.62 ± 0.02	48.75 ± 0.9	1.1 ± 0.2

Chapter 6

Conclusion

In this work we remove the independence assumption (treating parameters as if uncorrelated or remaining agnostic to parameter correlation) usually employed when regularizing the parameters of NNs. Since the benefit of a particular value for one parameter in an NN can only be determined in the context of the other parameter values, the independence assumption employed by common regularization techniques appears to be unjustified. To remove the independence assumption we create and evaluate a novel set of regularizers which account for the effect changing one parameter has on all other parameters in the network and adjust the regularization rate of that parameter accordingly. The creation of these regularization methods is the first contribution of this work. The regularizers adjust by either increasing (*Metric regularization*) or decreasing (*Selective L2*) the regularization rate based on the parameter's importance to the rest of the model. We experimentally verified the utility of each regularizer on a set of 5 datasets of varying complexity and interpretability. A key component of this work, and our second contribution, is to ensure that all of the novel methods are of the same time and memory complexity, relative to the number of parameters in the model, as *SGD*. This ensures that all of these methods remain practical.

The main, novel regularization method that we investigated in this work is the *Metric* regularizer. The theoretical analysis and motivation of this regularizer in Chapter 3 is our third contribution. We show that *Metric regularization* is the NN equivalent of *L2* regularization on linear regression models and results in the Minimum Mean Squared Error parameters being learned by the network. *Metric regularization* more acutely regularizes the parameters that are more impactful to the model's performance on training data by decreasing the Riemannian distance of the model parameters as opposed to the Euclidean distance. Thus, we calculate distance in parameter space but remain consistent with the notion of distance defined in the loss space. There is a common notion that the generalization gap between training and test performance of a model is due to the model learning minor, inconsistent variance in the training data labels and incorrectly attributing it to features of the input training data. In light of this notion, hindering the model from learning fine details of the training data should avoid the generalization gap from forming. Contrary to this intuition

on overfitting we find that the *Metric* regularizer does not improve the generalization of NNs.

Instead we find that the regularization methods that adaptively regularize the insignificant parameters of the model are able to improve generalization. This shows insight into what affects the generalizability of NNs and we find that it is not enough to indiscriminately reduce the norm of the model parameters, but rather to use as few significant parameters as possible. This reduces the norm of the model parameters, explaining the empirical correlation between the norm and generalizability, but most importantly does not hinder the model from learning as important parameters are still undisturbed. This challenges the notion that there is a trade-off between training and test data performance, similar to other works for example [Zhang *et al.* 2016 2018], and is the fourth contribution of this work.

This also sheds light on another observed phenomenon that different regimes of initialization exist. In this work we have called these regimes the small and large initialization regimes as in each regime the model parameters are initialized with either small or large values. It has been observed that the large regime overfits, while small regime initialization acts as an implicit regularization method [Geiger *et al.* 2020]. Intuition would suggest that it is merely due to the small weight regime resulting in models with small-norm parameterizations. If this were the case methods like *L2* would be able to replicate the generalizability of small regime models even when initialized in the large regime. As we show in our own experiments, this is not the case and *L2* is likely to over-regularize the model, hindering performance on training data, before it is able to meet the test data performance of the small regime models. Section 5.3 shows that by initializing in the small regime, a second implicit regularization emerges which is that only parameters that are necessary to fit the training data grow to be significantly greater than 0. As a result, small regime initialization allows the model to adaptively determine its own capacity and remain minimal in its use of parameters. In contrast we see that the large initialization regime results in the unnecessary model parameters lingering. While the model can learn to ignore these unnecessary parameters or use them to capture some of the variance in the training data, their lack of correspondence to stable input patterns means they likely capture noise in the training data that can result in errors arising in the test data. This insight into the different initialization regimes is our fifth and final contribution.

While challenging our intuition on overfitting and generalization, the experimental results of Chapter 5 also conflict with the theoretical results of Chapter 3. We see that despite *Metric regularization* having the MMSE property it fails to increase the test error performances of any of the models and consistently performs worse than all of the other regularization methods trained from the large initialization regime. We are left to interpret the possible reasons for this disconnect between our two primary results and contributions. The main assumption of the theory is that the model parameters follow a Multivariate Gaussian distribution and we used this as our prior distribution in deriving the *Metric* regularizer. This,

however, assumes that the second derivatives of an NN's parameters are constant, which is not necessarily true. The assumption of constant second derivatives of the parameters appears again in the use of the Laplace approximation, where a second-order Taylor expansion is used to approximate an exponential integral. If this assumption is incorrect, it has practical ramifications since the metric tensor is only locally defined. If this metric changes rapidly in parameter space, then it is difficult to regularize consistently and in a way that guides the model towards a beneficial area of parameter space. However, if small enough steps are taken in parameter space, then this is unlikely to be a problem and we would have observed less stability in the model performance if this had occurred. It is possible that ignoring higher-order derivatives results in too simple of a theoretical model to understand the complexity of an NN. This conclusion seems implausible, however, since we find that *L2 regularization* offers a fair regularization method while being derived from a more simplistic Gaussian prior. This does suggest that a possible direction of future work is to incorporate higher-order derivatives into our theoretical analysis and practical algorithms, although this will be computationally expensive. Our work with Metric regularization and Selective L2 also has relevance to Automatic Relevance Determination (ARD) priors [MacKay and Neal 1994; MacKay 1995]. Thus future work may aim to re-parametrize or adjust Metric regularization to penalise insignificant weights (in a "softer" manner than Selective L2) and, thus, draw more inspiration from ARD priors.

We may then look to another perspective of the prior other than whether it accurately reflects the nature of the model parameters. Namely the use of the prior to impart information or new restriction on the model which are not implicit in the data. This is the perspective more commonly associated to regularization. With this perspective it is apparent that the restriction from the Multivariate Gaussian, while more realistic, does not impart helpful information on the model. We see that this prior allows less variance away from 0 in the parameter dimensions that the data places stronger restrictions on and allows significant variance in the parameter dimensions that the data does not restrict. We see that this prior imparts the information that the data is not to be trusted, due to the risk of overfitting, and parameters that are more strongly determined by the data must be regularized. While intuitive, it is justified to conclude that this is just not useful information to impart based on our empirical results.

We could phrase the *Selective L2* regularizer in the Bayesian framework as a prior distribution with an independent or isotropic Gaussian prior only over the high-variance parameter dimensions. With this prior the parameters that are sufficiently restricted by the data have no prior distribution (equally phrased as a uniform or Jeffreys prior) and the parameters that are not restricted by the data we then impart the information that restricts them to be close to 0. It is also justifiable to treat these unimportant parameters as independent since they do not interact valuably with the other parameters in the model. We conclude that the Multivariate Gaussian, with the MMSE property, does not aid generalizability due

to its inability to impart new information in the model and in particular that high variance parameters should be kept close to 0. In a model where all parameters are necessary and the full capacity of the model is used to fit the data then the MMSE property would likely be more beneficial. In a model that we know to be over-parameterized it is necessary for us to impart this information at the expense of the MMSE property.

In conclusion, this work shows that it is necessary to consider the correlation between parameters of an NN when regularizing the model. This is due to the results showing the benefit of *Selective L2* over *L2 regularization* and *Selective L2*'s ability to reduce the generalization gap between the small and large regimes of training. We find that the aim should be to not regularize the parameters that are more useful given the context of the rest of the model parameters. Significantly, we see then that the data is capable of restricting the values of the correlated parameters in the network, and we are left to set restrictions on the uncorrelated dimensions of the parameter space exclusively. In this work we have presented a varying degree of informative priors, in the form of different regularizers, from uninformative uniform priors to very informative isotropic Gaussians. We conclude that with NNs the discussion is more nuanced and that certain dimensions of parameter space require varying degrees of informative priors. Fortunately, the geometry of the loss landscape provides this information and we are able to determine which dimensions to regularize and which dimensions to leave to be restricted by the data. Future work could likely seek to perform certain computations at a neuronal level which offer a similar implicit regularization effect to *Selective L2* regularization by dampening the effect of information coming from unhelpful connections in the network.

References

[Advani and Ganguli 2016] Madhu Advani and Surya Ganguli. Statistical mechanics of optimal convex inference in high dimensions. *Physical Review X*, 6(3):031034, 2016.

[Advani *et al.* 2013] Madhu Advani, Subhaneil Lahiri, and Surya Ganguli. Statistical mechanics of complex neural systems and high dimensional data. *Journal of Statistical Mechanics: Theory and Experiment*, 2013(03):P03014, 2013.

[Amari 2016] Shun-ichi Amari. *Information geometry and its applications*, volume 194. Springer, 2016.

[Baldi and Sadowski 2013] Pierre Baldi and Peter J Sadowski. Understanding dropout. In *Advances in neural information processing systems*, pages 2814–2822, 2013.

[Bansal *et al.* 2018] Yamini Bansal, Madhu Advani, David D Cox, and Andrew M Saxe. Minnorm training: an algorithm for training over-parameterized deep neural networks. *arXiv preprint arXiv:1806.00730*, 2018.

[Bayes 1763] Thomas Bayes. Lii. an essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfr s. *Philosophical transactions of the Royal Society of London*, (53):370–418, 1763.

[Brutzkus and Globerson 2019] Alon Brutzkus and Amir Globerson. Why do larger models generalize better? a theoretical perspective via the xor problem. In *International Conference on Machine Learning*, pages 822–830. PMLR, 2019.

[Carmo 1992] Manfredo Perdigao do Carmo. *Riemannian geometry*. Birkhäuser, 1992.

[Clevert *et al.* 2015] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[Dauphin *et al.* 2014] Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, pages 2933–2941, 2014.

[Dodson and Poston 2013] Christopher Terence John Dodson and Timothy Poston. *Tensor geometry: the geometric viewpoint and its uses*, volume 130. Springer Science & Business Media, 2013.

[Fisher 1922] Ronald A Fisher. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 222(594-604):309–368, 1922.

[Gauld 1974] David B Gauld. Topological properties of manifolds. *The American Mathematical Monthly*, 81(6):633–636, 1974.

[Geiger *et al.* 2020] Mario Geiger, Stefano Spigler, Arthur Jacot, and Matthieu Wyart. Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(11):113301, 2020.

[Geman *et al.* 1992] Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58, 1992.

[Glorot and Bengio 2010] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[Gur-Ari *et al.* 2018] Guy Gur-Ari, Daniel A Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. *arXiv preprint arXiv:1812.04754*, 2018.

[Hartigan 1998] JA Hartigan. The maximum likelihood prior. *The annals of statistics*, 26(6):2083–2103, 1998.

[Jaynes 1968] Edwin T Jaynes. Prior probabilities. *IEEE Transactions on systems science and cybernetics*, 4(3):227–241, 1968.

[Jaynes 2003] Edwin T Jaynes. *Probability theory: The logic of science*. Cambridge university press, 2003.

[Jeffreys 1946] Harold Jeffreys. An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 186(1007):453–461, 1946.

[Jost 2008] Jürgen Jost. *Riemannian geometry and geometric analysis*, volume 42005. Springer, 2008.

[Kaplan 1952] Wilfred Kaplan. *Advanced calculus*. Pearson Education India, 1952.

[Klambauer *et al.* 2017] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 972–981, 2017.

[Koenderink and Van Doorn 1992] Jan J Koenderink and Andrea J Van Doorn. Surface shape and curvature scales. *Image and vision computing*, 10(8):557–564, 1992.

[Krizhevsky *et al.* 2009] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Master's thesis*, 2009.

[Krogh and Hertz 1992] Anders Krogh and John A. Hertz. A simple weight decay can improve generalization. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 950–957. Morgan-Kaufmann, 1992.

[LeCun *et al.* 2010] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[Ly *et al.* 2017] Alexander Ly, Maarten Marsman, Josine Verhagen, Raoul PPP Grasman, and Eric-Jan Wagenmakers. A tutorial on fisher information. *Journal of Mathematical Psychology*, 80:40–55, 2017.

[Maas *et al.* 2013] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Citeseer, 2013.

[MacKay and Neal 1994] David JC MacKay and Radford M Neal. Automatic relevance determination for neural networks. In *Technical Report in preparation.* Cambridge University, 1994.

[MacKay 1995] David JC MacKay. Probable networks and plausible predictions-a review of practical bayesian methods for supervised neural networks. *Network: computation in neural systems*, 6(3):469, 1995.

[Porteous 2001] Ian R Porteous. *Geometric differentiation: for the intelligence of curves and surfaces*. Cambridge University Press, 2001.

[Rao 1945] C.R. Rao. Information and the accuracy attainable in the estimation of statistical parameters. *Bulletin of Calcutta Mathematical Society*, 37:81–91, 1945.

[Rissanen 1978] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

[Rumelhart *et al.* 1985] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning internal representations by error propagation*. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[Sagun *et al.* 2017] Levent Sagun, Utku Evci, V Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

[Skovgaard 1984] Lene Theil Skovgaard. A riemannian geometry of the multivariate normal model. *Scandinavian Journal of Statistics*, pages 211–223, 1984.

[Spivak 1970] Michael D Spivak. *A comprehensive introduction to differential geometry*. Publish or perish, 1970.

[Zhang *et al.* 2016] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[Zhang *et al.* 2018] Yao Zhang, Andrew M Saxe, Madhu S Advani, and Alpha A Lee. Energy–entropy competition and the effectiveness of stochastic gradient descent in machine learning. *Molecular Physics*, 116(21-22):3214–3223, 2018.