Examining applications of Neural Networks in predicting polygenic traits

McMaster University

Mu Tian

Date

Abstract

Polygenic risk scores are scores used in precision medicine in order to assess an individual's risk of having a certain quantitative trait based on his or her genetics. Previous works have shown that machine learning, namely Gradient Boosted Regression Trees, can be successfully applied to calibrate the weights of the risk score to improve its predictive power in a target population. Neural networks are a very powerful class of machine learning algorithms that have demonstrated success in various fields of genetics, and in this work, we examined the predictive power of a polygenic risk score that uses neural networks to perform the weight calibration. Using a single neural network, we were able to obtain prediction R^2 of 0.234 and 0.074 for height and BMI, respectively. We further experimented with changing the dimension of the input features, using ensembled models, and varying the number of splits used to train the models in order to obtain a final prediction R^2 of 0.242 for height and 0.0804 for BMI, achieving a relative improvement of 1.26% in prediction R^2 for height. Furthermore, we performed extensive analysis of the behaviour of the neural network-calibrated weights. In our analysis, we highlighted several potential drawbacks of using neural networks, as well as machine learning algorithms in general when performing the weight calibration, and offer several suggestions for improving the consistency and performance of machine learning-calibrated weights for future research.

Acknowledgements

First and foremost, I want to thank my amazing girlfriend Margaret for being so supportive throughout the entire research process. Secondly, I want to thank my supervisor, Dr. Canty, for all the support and insights offered over the course of the research, Dr. Paré and Shihong Mao for the the assistance in setting up the experiments, as well as Dr. Bolker for being a part of the defense committee.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Polygenic risk scores are scores used to assess an individual's risk of having a certain quantitative trait based on his or her genetics, and are widely used in precision medicine. Currently, a polygenic risk score is built from regression coefficients from an external *genome wide association study* (GWAS), sometimes, some type of adjustment is used to calibrate the risk score and to improve its predictive power. Machine learning is a widely popular set of computational methods aimed at solving big data problems, which have shown some promising results in calibrating polygenic risk scores to improve their predictive potential. Paré et al. (2017) showed that by using *Gradient Boosted Regression Trees* (GBRT) to adjust the weights of the polygenic risk score, they were able to achieve an improvement in R^2 over currently used methods of computing the polygenic risk score.

While Paré et al. (2017) achieved improved results in prediction R^2, over other methods, we hypothesize that it may be possible to further improve upon their results. Gradient boosting is a powerful class of machine learning algorithms, but with recent advances in *deep learning*, *neural networks* (NN) have the capability to achieve comparable, sometimes exceeding the predictive power of gradient boosted trees (De Brébisson et al., 2015; Guo and Berkhahn, 2016). In this work, we examine and compare the performance of a neural network calibrated polygenic risk score that incorporates recent advances and current best practices in the field.

1.1 Background and definitions

The study of genetics begins with the DNA. *Deoxyribonucleic acid* (DNA) is a molecule, found typically in the nucleus of a cell which carries the genetic information of the living organism. The genetic information stored in the DNA codes for cellular growth, function, and reproduction. DNA is made up of two strands of molecules called *nucleotides* coiled together to form a double helix structure. Each nucleotide is made up of one of the four types of nitrogen bases: adenine (A), thymine (T), guanine (G), and cytosine (C), combined with a sugar molecule and phosphate molecule. Due to the Watson-Crick pairing rules, which states that only adenine can be paired with thymine, and guanine with cytosine, each strand of the DNA contains the exact same genetic information, with different coding. A sequence of DNA that produces a protein or other functional element is referred to as a *gene*. The *genome* is the complete set of genetic information in an organism. Genomes are stored stored in *chromosomes*, which are made up of long strands of DNA tightly coiled together around histone proteins. There are typically 23 pairs of chromosomes in humans that make up the human genome (separate from the two strands of a single copy of a chromosome), receiving 1 copy of the chromosome from both parents. Due to genetic recombination, where the genetic material between chromosomes of the parents are exchanged, each copy of the chromosome in the child is not identical to the parent. *Single-nucleotide polymorphisms* (SNPs) are the most common form of genetic variation among humans. SNPs represents a change that occurs at a specific locus of the genome at a single nucleotide. For example, at a certain locus in the genome where most of the population have a C nucleotide, there is a subset of the population that instead have a T nucleotide, meaning there is a SNP at that locus of the genome. While most SNPs have no effect on the human body, certain genetic variations can have an effect on human health. SNPs are often studied to locate genes associated with diseases with a hereditary component, such as heart disease and schizophrenia. Most SNPs are *biallelic*, meaning that at a base position of the genome, there can be two possible nucleotide variations, however, there are SNPs that can have up to 4 possible bases at that location. In the previous example, the nucleotide can be C or T. The variations of the nucleotide are called the *alleles* at that locus. The common variation at a base position is called the *major allele*, and the less common allele is called the *minor allele*.

To analyze the effects of SNPs on various quantitative traits, studies are conducted using *genotype data*. A SNP genotype dataset of N individuals and M studied SNPs is represented as a matrix $X_{N \times M}$. Typically genotype data comes from a gene array where the columns represent the SNPs coded onto the array in its manufacture. SNP genotype data can take on values of $\{0, 1, 2\}$ to indicate the number of minor alleles in a single base pair. The SNP genotype data is used to test for association with an observable biological trait called the *phenotype*. The phenotypes of interest in this work are height, and body mass index (BMI).

SNPs are not entirely independent of one another. The frequency of one allele may be correlated with the frequency of alleles in other SNPs which are physically close to it on the genome. When the allele frequency at multiple SNPs is not independent, the SNPs are said to be in *linkage disequilibrium* (LD). SNPs in LD can can affect the association between that SNP and a measured trait. If two SNPs are in high LD, but only one SNP is a significant contributor to a phenotype, the study can incorrectly overestimate the effects of the non-contributing SNP on the measured phenotype due to the correlation between the allele frequencies. It is an important aspect in studies of associations between SNPs and phenotypes that the LD between SNPs are properly accounted for.

1.2 Genome Wide Association Study

To study the association between SNPs and a trait, a regression analysis is performed on the genotype data, and the regression coefficients are used to measure a linear association between each SNP and the phenotype. However, a problem arises with using traditional ordinary least squares (OLS) regression to model polygenic traits due to the high number of SNPs and comparatively low sample size, we run into the issue of $N \ll M$, meaning that the model would not have a unique solution. This issue is discussed at length in de Vlaming and Groenen (2015).

Widely adopted methodology to circumvent this issue is to carry out a *genome-wide asso-*

ciation study (GWAS). GWAS is typically conducted to test for association between SNPs and quantitative traits through regression analysis. In a GWAS, instead of regressing the trait of interest on all the SNPs at once (which has no unique solution due to $N \ll M$), GWAS performs the linear regression using each SNP one at a time, along with covariates such as design variables, gender, and non-genetic factors with a trait as the dependent variable. The resulting estimated regression coefficient is used as the association between the SNP and the trait. After estimating the regression coefficient, standard Wald tests are normally conducted to test the hypothesis that the true coefficient equals 0 to assess evidence of association between the SNP and the trait.

1.3 Applications of GWAS

Application of GWAS to identify associations between SNPs and phenotypes have been mixed, particularly in studies with low sample sizes. International Multiple Sclerosis Genetics Consortium (2010) performed a meta-analysis using two previously published GWAS on multiple sclerosis (MS), the International Multiple Sclerosis Genetics Consortium (IMSGC), and the Brigham and Women's Hospital (BWH) GWAS. Meta-analyses are used to combine the results from multiple studies on the same subject. In genetics, oftentimes issues can arise due to low sample, by combining the results from multiple studies into one meta-analysis, researchers can use the combined sample size to increase the statistical power of their studies. International Multiple Sclerosis Genetics Consortium (2010) was able to show that there existed a polygenic component to risk of multiple sclerosis (MS), but failed to identify any relevant loci contributing to the risk of MS. The study cites the lack of sample size in the meta-analysis as the main reason for the inconclusive result.

By contrast, meta-analyses of GWAS conducted on larger sample sizes have seen positive results. Allen et al. (2010) conducted a meta-analysis of GWAS data from across 61 studies encompassing 183,727 European subjects to test for the heritability of height as a polygenic trait. The study was largely successful, and researchers able to successfully identify many SNPs associated with height in over 180 loci, including over 100 novel loci. From the GWAS, the identified SNPs were able to explain approximately 10% of the variation in height. Speliotes et al. (2010) was one of the more successful studies conducted using GWAS meta-analysis to study associations between SNPs and BMI. The meta-analysis involved using the Genetic Investigation of Anthropometric Traits (GIANT) consortium for BMI (Locke et al., 2015), which is made up of 249,796 individuals all of European descent. Of the GIANT consortium, 123,865 of the subjects were used in the first stage study, and a follow up study was conducted on the rest of the 125,931 subjects. The study was able to confirm 14 loci previously associated with BMI, and discover 18 new loci, all with genome-wide significance level of 5×10^{-8}.

Chapter 2

Polygenic Risk Score

A *polygenic risk score* (PRS) is a quantitative measure that serves to predict the risk of a certain trait in individuals based on their genetics. Polygenic scores look at the relationship between an individual's genetic variation, represented by the SNP genotype data, and a certain phenotype. Once constructed, a polygenic risk scores can be used to predict the trait of an individual based on his or her genetics.

For a given phenotype, the polygenic risk score is defined to be a linear combination of a subset of the given coded genotypes of the SNPs, and their associated weights that reflect the association between each individual SNP and the phenotype. Let $\{x_{i,j}\}_{j=1}^{M_1}$ represent the genotypes of the M SNPs used in the PRS for individual i, the PRS is defined to be:

$$S_i = \sum_{j=1}^{M_1} x_{i,j} W_j$$

One of the most common ways to construct the weights of a polygenic risk score is to use the regression coefficients from an external GWAS conducted on the trait of interest as the weights in the risk score. Under the GWAS approach, the PRS is calculated to be:

$$S_i = \sum_{j=1}^{M} x_{i,j}\hat{\beta}_j \qquad .$$

Several problems with this GWAS approach were illustrated in de Vlaming and Groenen (2015). Firstly, due to the regression analysis done one SNP at a time, it will not result in a model that fits the optimal linear combination of SNPs to the desired polygenic trait. Secondly, regression done one SNP at a time does not account for SNPs in strong LD. Suppose that a region on the genome has one very high contributing SNP, and a number of non-contributing SNPs that are in high LD with the contributing SNP. Due to being in LD with the contributing SNP, the polygenic risk score could incorrectly over-emphasize the effects of the non-contributing SNPs on the measured trait. Thirdly, due to the size of M, it is computationally infeasible to fit any higher-order interaction terms, resulting in a model that does not take into account any possible interaction effects between SNPs that could either increase or decrease the effects of the interacting SNPs on the phenotype.

2.1 Pruning and Thresholding

Several improvements for calculating the polygenic risk score from the external GWAS model have been proposed to address some of the issues raised above. A p-value threshold could be set, and the model would only take SNPs from the GWAS that have association p-value less than the threshold. In doing so, many SNPs with no statistically significant association to the phenotype of interest could be discarded from the model in order to remove noise effects and improve precision. Common p-value thresholds range from 5×10^{-4} to 5×10^{-8} (Khera et al., 2018). Additionally, in order to reduce the effects of SNPs in LD, only the most significant SNP in a region is taken, and all other correlated SNPs within that region are discarded in the final model. When combined, this method is typically referred to as *pruning and thresholding* (P+T), and is one of the most widely adopted methods for constructing polygenic risk scores.

2.2 Applications of Polygenic Risk Scores

The first successful application of a polygenic risk score in humans using GWAS data was in International Schizophrenia Consortium (2009). Using the P+T method, the study first conducted extensive pruning of the SNPs independent of their association with the measured trait. After pruning the SNPs, the study tested setting the significance threshold at p-values of 0.1, 0.2, 0.3, 0.4, and 0.5. International Schizophrenia Consortium (2009) performed a case-control study of 3,322 cases and 3,587 controls of individuals of European descent, the study constructed a PRS using up to approximately one million SNPs associated with risk of schizophrenia. The best resulting PRS used a p-value threshold of 0.5, and was able to explain approximately 3% of the total variability in the risk of developing schizophrenia. The study also showed that the PRS for schizophrenia is also associated with risk of bipolar disorder, but not associated with non-psychiatric diseases such as coronary artery disease, Crohn's disease, or type 1 or type 2 diabetes. In addition, International Schizophrenia Consortium (2009) was the first to show in a simulation study that the effectiveness of polygenic risk scores increases as sample size increases, showing that the explained variance increases from approximately 3% to over 20% by simulating a sample of 20,000 case/control pairs.

Since then, studies have applied polygenic scores to several polygenic traits to test for both association and prediction of traits, with varying results. Simonson et al. (2011) aimed to use GWAS to identify common SNPs associated with cardiovascular disease (CVD) as well as to develop a polygenic score using the unadjusted GWAS regression coefficient estimates to predict for risk of developing CVD in the next 10 years. The study took genotype data from 1772 individuals from the SNP-Health Associated Resource (SHARe) after performing quality control. For testing the predictive power of the polygenic score, the sample was used in a 10-fold cross validation where each split used 9/10ths of the data to perform the GWAS while reserving the other 10% of the data as a testing set such that each of the 10 testing sets had no overlap in subjects. The results of the study was ultimately inconclusive, with no SNPs reaching the genome-wide significance level of 5×10^{-8}, while the polygenic risk score showed low predictive power, only explaining 1% of the variance in risk of CVD.

While many studies have used GWAS and polygenic scores, quite often the results were statistically inconclusive. Dudbridge (2013) showed that previous studies with mixed to negative results were not due to the limitations of the polygenic risk score, but rather could be attributed to studies having too few participants to achieve any substantial test power or predictive accuracy. Furthermore, Dudbridge (2013) went on to show that an increase in sample size is sufficient to improve both the power of association tests and the predictive accuracy of the polygenic risk scores. These conclusions serve to further reinforce the claims made in International Schizophrenia Consortium (2009) with regards to sample size which demonstrated that predicting quantitative traits will become more precise as sample data grows in size, observing an increase in explained variance from 3% to 20 %. However, the sample size required was orders of magnitude greater than what was available at the time, requiring 20,000 case control pairs compared to the original study of 3,322 cases and 3,587 controls.

Despite its prevalence in current construction of polygenic risk scores, the P+T method is not without its drawbacks. The main shortfall of P+T is that SNPs that are discarded from the GWAS for not being statistically significant enough could still contribute to the overall polygenic trait, resulting in a score with reduced predictive power. Simonson et al. (2011) noted in their study using polygenic models to predict CVD that "Thousands of small effects that are indistinguishable from background noise when performing a traditional GWAS can collectively account for a large proportion of the risk variation". Yang et al. (2010) was able to show that a model that takes into account all SNPs was able to explain a much larger proportion of heritability in human height as opposed to a model using only the SNPs that meet a certain significance value, showing that the most accurate polygenic scores should be built using all the SNPs simultaneously. The conclusion made by these studies is further reinforced by Ware et al. (2017) in a study conducted on the population-based longitudinal panel study Health and Retirement Study across four different polygenic traits: height, BMI, educational attainment, and depression. This study showed polygenic risk scores that include all available SNPs were able to either achieve a higher explained variance, or had non-inferior performance than a risk score that did not use all SNPs.

2.3 Penalized Regression Models

Using ridge regression to predict polygenic traits was first proposed in Whittaker et al. (2000). Ridge regression, originated in Hoerl and Kennard (1970), introduced an L_2 penalty term in the loss function of the model that penalizes large weights in the parameters. The L_2 penalty term incurs extra loss by summing the square of the regression coefficients. In OLS, the loss function is computed as:

$$J = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

By introducing the L_2 penalty term, the loss function becomes:

$$J_{L_2} = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{M} \beta_j^2$$

Where λ is a shrinkage parameter on the total sum of the squared regression coefficients. By minimizing this new penalized loss, certain parameters of the model should shrink close to 0 (but not equal 0), resulting in a model where certain SNPs will have regression coefficients close to 0. Unlike OLS, RR ensures that the model would have a solution even if $N \ll M$ (de Vlaming and Groenen, 2015).

Spiliopoulou et al. (2015) and de Vlaming and Groenen (2015) looked into the application of penalized regression models such as *ridge regression* (RR), *least absolute shrinkage and selection operator* (LASSO), and *elastic net* in polygenic risk scores. Using the penalized regression model, instead of regressing each SNP on the quantitative trait one at a time as with GWAS, the complete $N \times M$ matrix of genotype data is fitted to the response variable at once using RR, and its associated regression coefficients, $\hat{\beta}_{RR}$ are used as the weights of the PRS. From a simulation study conducted in de Vlaming and Groenen (2015), it was concluded that RR was able to slightly outperform the GWAS approach in genotype data with relatively simple genetic architecture using a relatively low sample size ($N < 10,000$). However, with an increase in sample size such as is present in available biobanks, de Vlaming and Groenen (2015) showed that penalized regression models could out perform GWAS in predictive power. In a simulation study done with 22,000 individuals and 500,000 SNPs RR achieved an increased prediction accuracy when compared to the GWAS approach, yielding a relative increase in prediction R^2 of approximately 6% on average. The simulation study also showed that in cases of $N \gg M$, RR does not yield a gain in prediction accuracy compared to GWAS.

LASSO regression (Tibshirani, 1996) is another form of penalized regression model that is closely related to RR, with the key difference being that LASSO uses an L_1 penalty term instead of the L_2 penalty used in RR. Using the L_1 regularization, the penalty term is calculated as the sum of the absolute values of the regression coefficients, so the loss function becomes:

$$J_{L_1} = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{M} |\beta_j|$$

This means that unlike RR, the regression coefficients in a LASSO regression can become exactly equal to 0, excluding certain SNPs from the model completely. By using LASSO, we end up with a sparse model when compared to RR (Spiliopoulou et al., 2015). Elastic Net (Zou and Hastie, 2005) is a combination of RR and LASSO. By taking a loss function that includes both the L_1 and L_2 penalties, this allows the linear model to solve $N \ll M$ problems while preserving the feature selection properties of LASSO regression. Because of the L_1 penalty imposed on the model, the regularization would result in a sparse model as well. Using LASSO regression (or its variants, such as the elastic net) has an advantage over RR in that LASSO regression models (and LASSO variants) are able to perform feature selection on SNPs by retaining key SNPs in the model and discarding SNPs that do not contribute to the overall variation in the tested polygenic trait. This could have positive effects on the resulting PRS as it removes some the SNPs that only contribute noise to the PRS. One downside of removing SNPs is that the model may also discard SNPs that do not contribute significantly to the measured trait, but could have a significant effect on the trait when added together.

Warren et al. (2014) examined the predictive accuracy of polygenic risk scores when compared to penalized regression models using all SNPs for predicting low-density lipoprotein (LDL) and high-density lipoprotein (HDL) cholesterol. Using genotype data from the Whitehall II study and Women's Health and Heart Study, with a total sample size of 7,432 individuals and performing a 10-fold cross validation, they concluded that sparse models, such as LASSO, performed better than RR in predictive accuracy. However, sparse models did not have any statistically significant improvement in performance when compared to polygenic scores constructed using the P+T method, and all models still had weak predictive performance. In the discussion, the study cited Dudbridge (2013) and concluded that the predictive accuracy of the models is limited by the small sample size, and suggested future studies with larger sample sizes and different phenotypes should be used for proper model comparison.

2.4 LDpred

The most prevalent issue in the pruning and thresholding method of modeling polygenic risk scores is that the model does not take into account the LD between SNPs, leaving out valuable information from the final score and negatively affecting predictive accuracy. Vilhjálmsson et al. (2015) introduced a novel method of constructing polygenic risk scores called *LDpred* that improves upon the pruning and thresholding method. For a genotype matrix $\mathbf{X}_{N\times M}$ and $\mathbf{y}_{N\times 1}$ a vector of phenotype responses, LDpred takes a Bayesian approach to modeling polygenic scores by first assuming a point-normal mixture prior for the distribution of the the effects of the SNPs given by:

$$\beta_j \overset{iid}{\sim} \begin{cases} N\left(0, \dfrac{h_g^2}{Mp}\right) & \text{with probability } p \\ 0 & \text{with probability } 1-p \end{cases}$$

Where h_g^2 is explained heritability of genotyped variants, estimated from GWAS summary statistics accounting for LD and sampling noise, and p is the probability that a SNP is drawn from a Gaussian distribution. To estimate the posterior mean effect, the paper assumes distant markers are uncorrelated, and the posterior mean effects within a region ℓ for $\mathbf{X}$ is given by:

$$E(\beta^{\ell}|\tilde{\beta}^{\ell}, D) \approx \left(\frac{M}{Nh_g^2}I + D_{\ell}\right)^{-1}\tilde{\beta}^{\ell}$$

Where D is the matrix of pairwise LD between the SNPs in the data. β^{ℓ} is regression coefficient from OLS within region ℓ, $\tilde{\beta}^{\ell}$ is the least squares estimate of β^{ℓ}, and D_{ℓ} is LD matrix between SNPs in region ℓ. Due to difficulties analytically deriving the posterior mean under the mixture prior assumed, the posterior mean is estimated using the Gibbs Sampler. The PRS is computed by using the estimated posterior mean effects of each SNP as the weight.

By taking into account the LD information in the construction of the polygenic score, the LDpred method was able to achieve an increase of prediction R^2 in both simulation and application to real datasets. It is detailed in Vilhjálmsson et al. (2015) that in application to six quantitative traits: schizophrenia, multiple sclerosis, breast cancer, type 2 diabetes, coronary heart disease, and height, LDpred achieved a much better prediction R^2 compared to P+T, ranging from 11% to 30% relative increase in R^2. It was concluded that polygenic risk scores from LDpred improves upon P+T because the LD pruning method prunes all SNPs in LD with the selected SNP, thereby discarding any potentially relevant SNPs from the polygenic score. By contrast LDpred is able to model for the effects of SNPs in LD, which yielded additional gains in prediction accuracy compared to P+T.

While LDpred was able to achieve a significant improvement over P+T, Vilhjálmsson et al. (2015) noted that the method still has several limitations. The LDpred approach, like P+T, still relies on summary statistics obtained from previous GWAS, meaning the predictive power of LDpred is reliant on how well the GWAS represents the target population. Another issue raised in Vilhjálmsson et al. (2015) was that the point-normal mixture prior distribution used in LDpred may not be an accurate representation of the true genetic architecture.

2.5 Machine Learning in Predicting Polygenic Traits

Machine learning algorithms are a class of statistical models that are capable of learning complex nonlinear models based on structure learned from the data. Several machine learning algorithms have the capability of outperforming linear models in terms of predictive power, and thus it is proposed that using more powerful machine learning algorithms to perform a calibration on the weights of a polygenic risk score could lead to an improvement in its predictive power. Paré et al. (2017) was the first instance that we could find in literature of successfully utilizing *gradient boosted regression trees* (GBRT) to build a calibrated polygenic risk score from an external GWAS that achieved higher predictive powers than previous methods. Paré et al. (2017) introduced two novel approaches to calibrating the polygenic risk score from the external GWAS. First, the paper introduced using a machine learning algorithm to calibrate the PRS to a target population through the use of a *calibration set.* Paré et al. (2017) proposes taking a sample from the target population performing a GWAS, then training a machine learning algorithm to fit a model to the difference between the the regression coefficients from the external GWAS ($\hat{\beta}_{ext}$), and the regression coefficients derived from a GWAS conducted on the calibration data ($\hat{\beta}_{obs}$). The difference between the two estimated regression coefficients is computed to be

$$d = (\hat{\beta}_{ext} - \hat{\beta}_{obs}) \times \text{sign}(\hat{\beta}_{ext})$$

The data is then split into 5 contiguous parts to avoid potential LD spillover. A machine learning algorithm, $h(x)$, is fitted to the 4 of the 5 sets of the data $D = \{|\hat{\beta}_{ext}|, d\}$, using $|\hat{\beta}_{ext}|$ as the input feature (the covariate), and d as the dependent variable. Next, the machine learning algorithm uses the remaining set of the data to predict an adjustment factor $\hat{d}$ for that set, and the process is repeated to predict the $\hat{d}$ for each of the 5 sets in the split, using the remaining 4 sets of data to train the model (while holding model parameters the same). The predicted $\hat{d}$ is then used to calibrate the $\hat{\beta}_{ext}$ from the external GWAS to obtain the calibrated weights, $\hat{w}$. The machine learning calibrated weights was then calculated as:

$$\hat{w} = (|\hat{\beta}_{ext}| + \hat{d}) \times \text{sign}(\hat{\beta}_{ext})$$

The formula presented in this work for $\hat{w}$ differs from the formula presented in Paré et al. (2017). This is due to the fact that we discovered an error in the formula presented by the original paper. A detailed report on the erroneous formula is presented in detail in Section 4.3.

Paré et al. (2017) primarily focused on the utilization of gradient boosted regression trees to perform the calibration of the weights. Boosting is a class of meta-algorithms that creates a powerful learner from an ensemble of weaker learners, such as Decision Trees. Unlike the Random Forest, where each decision tree is fitted to the data independent of each other, boosting seeks to improve prediction results by fitting each decision tree in sequence, where the next decision tree in the sequence is fitted to correct for poor predictions made by the previous iteration of the model. *Gradient Boosted Trees* (Friedman, 2001) is a very popular and powerful form of the boosting algorithm, and has been successfully utilized in many areas of predictive analytics. The GBRT was trained using two external GWAS as the source of the $\hat{\beta}_{ext}$ in order to build two polygenic risk scores for the traits of interest. The Genetic Investigation of Anthropometric Traits (GIANT) (Wood et al., 2014) consortium summary association statistics were used to train the model for height, and the summary association statistics from GIANT consortium for BMI (Locke et al., 2015) was used for BMI. The external summary statistics for 1.98M SNPs taken from the GIANT consortium was based on approximately 220k sample subjects. GWAS was performed on the genotype data, with height and BMI as the measured phenotypes, to obtain the external regression coefficients.

The calibration and validation data comes from the *UK Biobank* (UKB) (Sudlow et al., 2015). To remove any potential effects of outliers, individuals with phenotypes measured at less than the 1st and greater than the 99th percentile were removed from the data. The UKB data was based on genotype data of 152,249 subjects across the UK, and 140,215 individuals of European (British and Irish) Caucasian ancestry were used in the dataset after performing quality control and removing outliers. From the UKB dataset, 10,000 individuals were randomly sampled from the data to be used for calibration set, and then GWAS was performed on the 10,000 individuals to calculate $\hat{\beta}_{obs}$. Final validation dataset used to test the predictive power of the calibrated PRS for height and BMI consisted of the remaining 130,215 individuals not used for the machine learning calibration. To compute the $\hat{\beta}_{obs}$, a GWAS was conduced on the calibration dataset, using the SNP data as well as the sample population's age and sex as covariates for the GWAS.

The second key contribution Paré et al. (2017) made to the development of polygenic risk scores is a novel way to adjust for the linkage disequilibrium in the polygenic risk score, dubbed *LD adjustment*. LD adjustment differs from the P+T approach of building polygenic risk scores in that instead of pruning the SNPs in LD, the correlated SNPs are down-weighted to reflect their pairwise correlation to neighboring SNPs. Let $r^2_{j,k}$ represent the pairwise LD/correlation (r^2) between the j-th and k-th SNPs, then the LD adjustment value is calculated as

$$\eta_j = \sum_{k=j-100}^{j+100} r^2_{j,k} \qquad .$$

A distance of 100 SNPs was used in the calculation of the LD adjustment value as it was assumed in Paré et al. (2017) to be sufficient to model the linkage disequilibrium. The LD adjusted weight for SNP j used in the polygenic risk score is then

$$\tilde{W_j} = \frac{W_j}{\eta_j} \qquad .$$

The resulting PRS is used to calculate the prediction R^2 when applied to the validation sample of the remaining 130,215 individuals in UKB.

In the polygenic risk score for height, the method described in Paré et al. (2017), called GraBLD, was able to achieve a prediction R^2 of 0.239, improving upon the prediction R^2 of

LDpred (0.207), Pruning and Thresholding (0.220), and the unadjusted GWAS (0.165). In the polygenic risk score for BMI, GraBLD was able to obtain a prediction R^2 of 0.082, once again beating LDpred (0.074), Pruning and Thresholding (0.069), and the unadjusted GWAS (0.069).

2.6 A Neural Network approach to calibrating weights

While GBRT was the most successful algorithm, and primary focus of Paré et al. (2017), several other machine learning algorithms were tested in the study such as *support vector machines* (SVM), *random forests* (RF), and neural networks (NN). We noted that the package used in Paré et al. (2017) to build the neural networks (*nnet*) is limited when it comes to building neural networks, and lacks implementation of many key features of fitting neural networks today. In this paper, we introduce a different neural network approach to calibrating the weights. By utilizing modern advances in deep learning and training neural networks, we believe that it may be possible to improve on the results obtained by the GBRT calibrated PRS by using a neural network to calibrate the weights instead.

Our goal is to examine whether or not neural networks, when combined with modern deep learning best practices, can be as effective as GBRT when it comes to calibrating the weights, and whether or not the resulting NN calibrated PRS will have comparable predictive powers. To have a fair comparison across the experiments, we aim to keep the methods used in in our experiments as close to Paré et al. (2017) as possible, including using the same external GWAS data, the same calibration and validation data, as well as performing the LD adjustment to the NN calibrated weights. In doing so, we hope to fully examine the predictive capabilities of the PRS after performing the neural network based calibration.

Chapter 3

Neural Networks - Background

Neural networks are an extremely powerful class of algorithms that first drew inspiration from the human brain. Neural networks have demonstrated strong success in various fields of modern machine learning tasks via *deep learning* (LeCun et al., 2015). Deep learning allows for increasingly more complex neural networks, and as a result, allows for the model to discover increasingly non-linear relationships between the input features and labels (Eldan and Shamir, 2016). As demonstrated in various machine learning competitions such as Kaggle competitions, NN showed the potential to outperform GBRT in supervised regression tasks (De Brébisson et al., 2015; Guo and Berkhahn, 2016). We hypothesized that by applying recent advances in techniques to optimize model fitting, neural networks might be able to outperform GBRT in calibrating the weights of polygenic risk score to improve its predictive capabilities.

3.1 Structure

The structure of a neural network is composed of an *input layer, output layer,* and *hidden layers*. Given $X = \{\mathbf{x_i}, y_i\}_{i=1}^{N}$ where each $\mathbf{x_i}$ is an $M \times 1$ vector representing M features in each data point, and y_i represents either the corresponding class the data belongs to for classification tasks or a response value in the case of regression models. Each layer of the neural network is made up of *perceptrons*, often referred to as *hidden units* or *neurons*. Each layer in the neural network is connected to the successive layer by weights W that are initialized according to

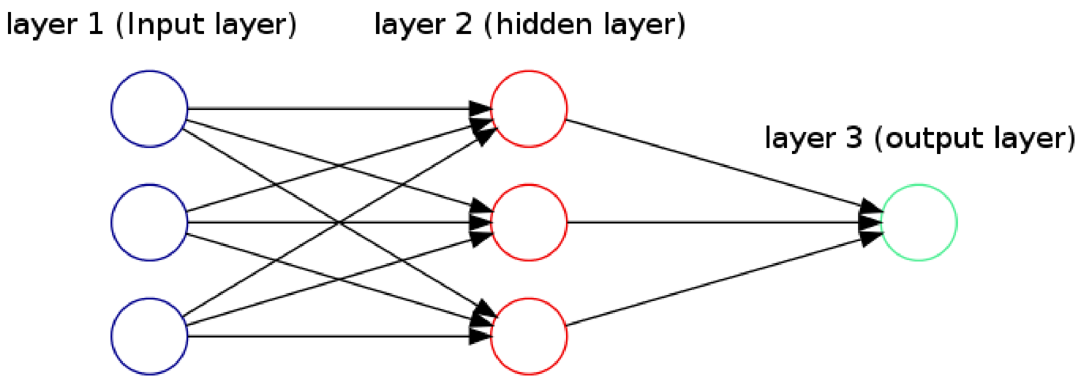


Figure 3.1: Diagram showing a feed forward neural network with 1 hidden layer. The features that make up the input layer of the neural network are propagated forward via the connecting weights (depicted by the arrows) to the hidden layer, which are made up of perceptrons. The calculated outputs from the hidden layers are then used as the inputs for the output layer. Source: tgmstat.wordpress.com (Used with permission of the Author: Thiago G. Martins)

some pre-specified method. The weights in the neural network are iteratively updated during the model training process in order to improve the performance of the network.

Let M_0 represent the number of features in X, and M_i represent the number of perceptrons in the i-th hidden layer in the neural network, and $\mathbf{x^i}$ represent the values of the perceptrons in the i-th layer. In the input layer, the $M_0 \times 1$ vector of features for each entry of data $\mathbf{x_i^0}$ is passed to each of the perceptrons in the first hidden layer of the neural network via weights that connect the current input layer to the first hidden layer in the neural network, $W_{0,1}$ which is a $M_1 \times M_0$ matrix, by the matrix multiplication $W_{0,1}^T \mathbf{x_i^0}$. More generally, the weights connecting the i-th hidden layer to the j-th hidden layer W_{ij} is of dimension $M_j \times M_i$ such that every perceptron in layer i is mapped to every perceptron in layer j. An output value of the j-th hidden layer is computed based on the perceptrons in that layer, and that output becomes the input value to the next (k-th) hidden layer. The data is passed through the neural network sequentially through each hidden layers until the final output layer. The output layer is the final predicted value of the neural network. The number of hidden units can depend on the task the network is performing. In the case of regression, there is only one unit in the output layer.

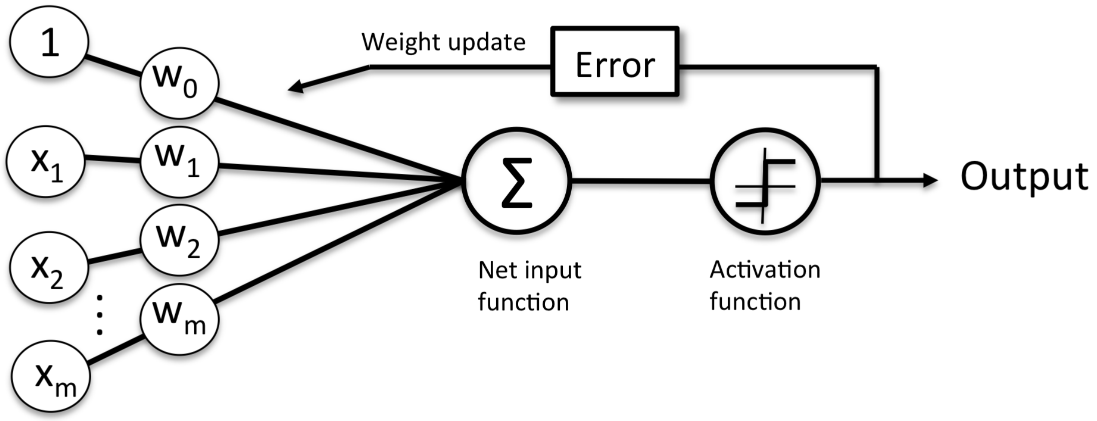


Figure 3.2: Diagram depicting the architecture of a perceptron, which are make up the hidden layers of the neural network. The perceptron consists of a weighted sum of the input weights (Net input function), and a non-linear activation function of the weighted sum. Source: Raschka (2015) (Used with author permission)

3.2 Perceptron

Perceptrons are the building blocks of neural networks. A perceptron is made up of two parts, the *weighted sum* z and *activation function* $\sigma(z)$. The weighted sum takes a sum of the values of the outputs from the previous layer multiplied by the connecting weights to that perceptron. Consider inputs into layer j $\mathbf{x^j} \in R^{M_j}$ with corresponding weights $W \in R^{M_j}$ that connect the previous layer of units to that perceptron, then the weighted sum of the perceptron would be $z = \sum_{i=1}^{M_j} x_i^j W_i = W^T \mathbf{x^j}$. The activation function takes the weighted sum and performs a non-linear transformation $\sigma(z)$ in order to obtain the output value for that perceptron. The most common activation functions in a neural network are: *sigmoid, TanH,* and *Rectified Linear Units* (ReLU) (Nair and Hinton, 2010). The definitions of the common activation functions are listed in Appendix A.1.

For easier modification to the neural network architecture, it is often easier to separate the neurons in a layer into two separate layers, feed-forward layer and activation layer. Doing so allows for the freedom of adding other functions between the core layer and the activation, or to change the order of functions within the layer.

3.3 Model Training

In neural networks, the model is trained to minimize a specified *loss function* $J(W)$ where W is all the weights in the network. The loss function is used to calculate how much the model predictions $\hat{y}$ deviate from the true values y, and is then used to update the weights in the neural network. Common loss functions used in neural networks and their formulas are listed in Appendix A.2.

Neural networks are trained iteratively, and each iteration is split into two phases, the *forward pass* and *backward pass*. In the *forward pass*, the features of the dataset are input through the neural network in order to obtain a predicted value, $\hat{y}$ in order to calculate the loss function, $J(W)$.

In the backward pass phase of the training process, the *gradient* of the loss function at the k-th iteration, $\nabla J(W) = \frac{\partial}{\partial W^{(k)}} J(W)$ is computed, and then used to update the weights for the next iteration.

Most neural networks compute $\nabla J(W)$ using a method known as *backpropagation* (Rumelhart et al., 1986). Backpropagation is an efficient method of computing the gradient by utilizing the chain-rule of calculus where instead of calculating the loss with respect to W, the gradient of the error in one layer is first calculated, then "propagated" back to the previous layer to compute the gradient of that layer.

let $\mathbf{x^i}$, $\mathbf{x^j}$ be the values of the hidden units in the i-th and j-th layer, respectively, let W_{ij} be the weights connecting $\mathbf{x^i}$ to $\mathbf{x^j}$, and let $z_j = W_{ij}^T \mathbf{x^i}$ (the weighted sum of layer j before the activation). Using the chain rule we can see then:

$$\frac{\partial}{\partial W_{ij}} J(W) = \frac{\partial J(W)}{\partial z_j} \frac{\partial z_j}{\partial W_{ij}} = \frac{\partial J(W)}{\partial z_j} \mathbf{x^j}$$

let $\delta_j = \dfrac{\partial J(W)}{\partial z_j}$ and let layer k be the layer after layer j. Using the chain-rule again, we can get

$$\begin{aligned}\delta_j = \frac{\partial J(W)}{\partial z_j} &= \sum_k \frac{\partial J(W)}{\partial z_k}\frac{\partial z_k}{\partial z_j} \\ &= \sum_k \delta_k \frac{\partial z_k}{\partial \mathbf{x^j}}\frac{\partial \mathbf{x^j}}{\partial z_j} \qquad .\end{aligned}$$

Because $z_k = W_{jk}^T \mathbf{x^j}$, then $\dfrac{\partial z_k}{\partial \mathbf{x^j}} = W_{jk}$, and since $\mathbf{x^j} = \sigma(z_j)$, $\dfrac{\partial \mathbf{x^j}}{\partial z_j} = \sigma'(z_j)$, and so

$$\begin{aligned}\delta_j &= \sum_k \delta_k W_{jk} \sigma'(z_j) \\ \frac{\partial}{\partial W_{jk}} J(W) &= \delta_k \mathbf{x^j}\end{aligned}$$

Let the k-th layer of the neural network be the output layer. In the case of regression, the output layer consists of one perceptron with no activation function. In this case, we have $z_k = \hat{y}$, so then $\delta_k = \dfrac{\partial J(W)}{\partial z_k} = \dfrac{\partial J(W)}{\partial \hat{y}}$.

By using backpropagation, the gradients for each hidden layer are calculated, which can now be used to update the weights by using an *optimization algorithm* in order to reduce the loss.

3.4 Optimizing neural networks

Optimization algorithms describes how to update the weights using the calculated gradient. The choice of optimization algorithm can play a huge part in how quickly the overall model fitting converges as well as the fitted model's predictive power. The most basic method of optimization is *gradient descent* where the weights are updated according to:

$$W^{(t+1)} = W^{(t)} + \alpha \nabla J(W)$$

Where α is a pre-specified learning rate. A popular variant of the gradient descent algorithm is *stochastic gradient descent* (SGD) (Robbins and Monro, 1951; Bottou, 2010) where instead of the whole data being used to train the network in each iteration, only one sample randomly selected from the data is used to calculate the loss and perform the weight update.

Consider the loss function $J(W)$. Since $J(W)$ is total loss of the model, then it can be written as the sum of the loss of each predicted value, i.e. $J(W) = \sum_i^N J_i(W)$ where $J_i(W)$ represents the calculated loss of a single prediction. Thus, using SGD, the weights are updated according to:

$$W^{(t+1)} = W^{(t)} + \alpha \nabla J_i(W)$$

By using one point of data instead of the entire data to perform the weight update at each iteration, SGD adds noise to the gradient descent process that would normally be smoothed out over all errors, thus reducing the likelihood of overfitting to the training data and improving model generalization. Additionally, by only needing to calculate the forward pass of one point as opposed to the entire data before computing the gradient, SGD is able to speed up the rate of model convergence while retaining a good approximation of the global minimum (Ng, 2000).

3.5 Batch Training

Instead of training over all data at once, neural networks typically divide the training data into subsets of the data, called *batches.* Through training using batches, the weight is updated according to the error of each batch instead of the overall error. All batches are used to train the neural network, with an *epoch* denoting one cycle where all batches have been used to train the network. Performing gradient descent on batches instead of the entire dataset is called *Batch Gradient Descent* (BGD). When compared to SGD, training using BGD results in less noise than using a single data point to update the model, and can lead to less drastic weight updates. A commonly used compromise between BGD and SGD is *mini-batch gradient descent*, where one sample from each batch is used to update the weights in the backward pass. Since neural networks are typically trained in batches, SGD and mini-batch gradient descent are often interchangeable.

3.6 Adaptive Optimizers

The choice of the learning rate α can be crucial to the the model training process. Choosing too high a learning rate can lead to the model not being able to converge to a minimum, while too low a learning rate can significantly slow down the convergence rate, and carry the risk of the model being stuck in a local minimum instead of achieving the global minimum. This problem can addressed through the implementation of *momentum* and *adaptive learning rate* in the optimization algorithm. Momentum makes use of the gradient updates in previous iterations in addition to the current gradient to perform the weight update, and adaptive learning rates change the learning rate α based on the gradient at that iteration. Some of the most popular optimizers that make use of momentum and adaptive learning rates include: *SGD with Momentum* (Qian, 1999), *SGD with Nesterov Momentum* (Nesterov, 1983; Sutskever et al., 2013), *adaptive moment estimation* (Adam) (Kingma and Ba, 2014), and *RMSProp* (Tieleman and Hinton, 2012). Definitions of these optimization algorithms are listed in Appendix A.3. In Wilson et al. (2017), it was noted that although adaptive optimization algorithms such as Adam and RMSProp did lead to a faster model convergence rate when compared to SGD and SGD with Nesterov Momentum, the final model often performed worse in validation and test error in certain prediction tasks.

Algorithm 1 Weight update algorithm for a k-layer NN using Gradient Descent

```
1: Initialize weights $W$ for the model
2: for $t$ from 1 to $T$ epochs do
3:     Input features $\mathbf{x^0}$ to the input layer,
4:     The outputs are passed forward as the inputs for the successive layer
5:     Once $\mathbf{x}$ reaches the output layer (layer $k$), compute the loss, $J(W)$ for each output unit
       and compute $\delta_k$
6:     for $i$ from $k-1$ to 1 do
7:         Compute $\delta_i = \sum_j \delta_j W_{ij} \sigma'(z_i)$ where $j = i + 1$
8:         Compute $\frac{\partial}{\partial W_{ij}} J(W) = \delta_j \mathbf{x^i}$
9:     Weight Update $W_{ij}^{new} = W_{ij}^{old} - \alpha \frac{\partial}{\partial W_{ij}} J(W)$
```

3.7 Model Initialization

Initialization of the weights in the neural network can have profound effects on the speed at which the loss function converges to a global minimum. Similar to numerical optimization methods like the *Newton-Raphson Method*, proper choice of weight initialization can speed up the model convergence rate, and ensure that the model does not get stuck in a local minimum. Modern initialization methods such as the *Glorot Initialization* method (Glorot and Bengio, 2010) have been shown to improve the convergence rate of the model. The Glorot Initialization samples the weights from a normal distribution with mean zero, and variance of $\frac{2}{n_{in} + n_{out}}$ where n_{in} is the number of neurons in the previous layer, and n_{out} is the number of neurons in the connecting layer. An updated initialization method, the *Kaiming Initialization* in He et al. (2015) proposes sampling from a normal distribution with mean 0 and variance of $\frac{2}{n_{in}}$. Using the new initialization method, He et al. (2015) managed to achieve a faster convergence rate when compared to the Glorot Initialization method, particularly for deep neural networks.

3.8 Neural Network Regularization

Due to the complexity of neural networks, control for overfitting is often a key concern. Several methods have been devised to curb overfitting and improve the generalizability of the neural network.

Dropout (Srivastava et al., 2014) is a very useful form of regularization for neural networks. Dropout works by "dropping" hidden units from the model during the forward pass by setting the output value of the hidden unit to 0 with a fixed probability p, along with any weights connecting to that unit. Dropout works by introducing additional noise to the model via the dropped neurons, thus reducing overfitting. Additionally, dropout can be seen as a form of *model averaging* where the resulting models after applying dropout can be viewed as a new "thinned" neural network with a different network architecture. The resulting prediction from the trained model can thus be viewed as an average prediction of many different neural networks. By combining the predictions of many models, erroneous predictions from one model can be averaged out by multiple good predictions made by many other models (LeBlanc and Tibshirani, 1996; Güneş et al., 2017).

L_1 and L_2 regularization can be utilized in neural networks to reduce the variability of the predictions (Ng, 2004). L_1 and L_2 regularization are useful regularization methods that are used in many machine learning models. This form of complexity control works by introducing a penalty term on the number of parameters in the model, thus penalizing the model for having too many parameters. In neural neural networks, as is with RR and LASSO, L_1 and L_2 regularization is added to the loss function as:

$$\text{Loss}_{L_2} = J(W) + \lambda \sum W^2$$
$$\text{Loss}_{L_1} = J(W) + \lambda \sum |W|$$

Where λ is a constant scaling factor and W is the collection of all weights in the neural network.

Weight decay (Krogh and Hertz, 1992) is another form of regularization that imposes a penalty on higher weights in the neural network. Unlike L_1 and L_2 regularization, weight decay is calculated during the weight update stage of the backward pass. The weight decay function is written as:

$$W^{(t+1)} = W^{(t)} + \alpha \nabla J(W^{(t)}) - \lambda W^{(t)}$$

In neural networks, L_2 regularization is often implemented interchangeably with weight decay. However, there is a subtle difference that could affect the overall performance of the model. If we take the gradient of the L_2 loss, we would get a weight update function of

$$W^{(t+1)} = W^{(t)} + \alpha \nabla J(W^{(t)}) - 2\lambda W^{(t)}$$

due to the quadratic term, resulting in a heavier penalty for the weights compared to weight decay.

Batch normalization (Ioffe and Szegedy, 2015) (batchnorm) is an algorithm that is typically applied to the weighted sums of the neurons z before applying the activation function, and is commonly used in optimizing the neural network training process. While the primary purpose of batchnorm is for network optimization by allowing for higher learning rates for optimization algorithms (hence speeding up the convergence rate) and reducing model sensitivity to the initial weights, it can act like a regularizer by introducing noise into the model and can "in some cases, eliminate the need for dropout" (Ioffe and Szegedy, 2015). The batchnorm algorithm, as well as the derivation of its gradients for backpropagation, is listed in Appendix A.5.

Early stopping (Prechelt, 1998) is an effective technique to prevent overfitting by terminating the training process once the model performance stops improving. To measure model performance, the training dataset is further split into a training and validation set, the model is fitted to the training set, and its performance measured on the validation set. Once the measured validation loss of the model stops decreasing over a specified number of epochs, the training terminates. Since the model stops training once the specified conditions are met rather than training through the full number of epochs, this method has the potential to stop neural networks from overfitting, and can speed up training time. One downside of using early stopping is that the model may terminate too early and it may have found a better solution if allowed to train through the full number of epochs. Another way to control for overfitting would be through the use of *model checkpoint*. Unlike early stopping, model checkpoint lets the neural network train through the full number of epochs, but saves the network weights whenever a new minimum in validation loss is achieved. Once the network is fully trained, the weights of the best performing model can be loaded into the network.

3.9 Learning Rate Scheduling

Learning rate scheduling refers to changing the learning rate throughout the training process in a pre-specified manner. Typically, higher learning rates enable the NN to converge faster, but too high a training rate and the algorithm may not converge to a minimum of the loss function. Conversely, a low training rate can ensure the network converges, but a low learning rate slows down the model training process and runs the risk of getting stuck in a local minimum instead of the global minimum. A typical process for getting the most out of the benefits offered by higher and lower learning rates is to start off with a higher learning rate, then gradually decrease it over the training epochs. This method allows for the NN to fully explore the loss surface and avoid becoming stuck in a local minimum in the early phases of training, while decreasing the learning rate in later phases allows the model to converge to a global minimum. Instead of using a strictly decreasing learning rate schedule, Smith (2017) proposed the use of *cyclical learning rates*. In a cyclical learning rate, the learning rate is increased and decreased in a cyclical fashion over the course of the model fitting process. By allowing the learning rates to increase to higher values multiple times during model training, a cyclical learning rate schedule will allow for the model to explore the loss surface much better than a strictly decreasing learning rate schedule, thus resulting in faster training and improved model performance. Smith (2017) showed that using a cyclical learning rate outperforms traditional learning rate schedules for neural networks in both training speed and predictive power.

In this work, we utilized two possible learning rate schedules for our experiments. Firstly, we start off with a higher learning rate, then decrease it after 5 epochs of no improvement in validation loss. Another learning rate schedule we used was a step-wise learning rate scheduler, where the learning rate decays by a set factor after a certain number of training epochs.

3.10 Gradient Clipping

One issue with performing regression tasks using neural networks is the possibility of exploding gradients. Exploding gradients occur when the predictions made by the neural network are far from the target values, which leads to very large loss gradients and weight updates. This can cause the neural network to become unstable and result in a situation where the weights in the neural network increase to infinity. The increase of the weights to infinity would result in an overflow error in the code, causing weights to become "Not a Number" (NaN) in our code. One way to address the issue of exploding gradients is to apply *gradient clipping*, where a threshold is set at a certain value such that any larger gradients would be cut off at the threshold value. Pascanu et al. (2013) introduced some practical applications for clipping the gradients based on empirical experiments. Pascanu et al. (2013) suggested basing the clipping value on the average of the norm of the gradient of the batches across a sufficiently large number of samples, and notes that from their experiments, a range of clipping values around the norm can still result in model convergence. However, as the model complexity hyperparameters change, the clipping value required to prevent exploding gradients may also change.

3.11 Neural Networks in Genetics

Deep learning has previously been underutilized in genetics due to the high computational costs and lack of sufficient sample size in data. In recent years, deep learning had seen a surge in application to genetics due to the rise of biobanks, which allowed for data with significantly large sample sizes, as well as improvements in computational power which allowe researchers to experiment with more computationally intensive network architectures.

Montañez et al. (2018) was the first paper to combine feed-forward neural networks with previously established GWAS results to further enhance the predictive powers of significant SNPs associated with obesity. The study took the results of an external GWAS that was performed between individual SNPs and obesity under an additive logistic regression model. Four subsets of SNPs with p-values corresponding to less than 1×10^{-5}, 1×10^{-4}, 1×10^{-3}, and 1×10^{-2} were used to train a feed-forward neural network to classify obesity. The study showed promising results in that a neural network trained using SNPs with a p-value less than 1×10^{-2} (2465 SNPs) was able to better predict cases of obesity than simply using SNPs suggestive of association in the additive logistic regression model with p-value less than 1×10^{-5} (5 SNPs). By using neural networks, the model was able to capture the effects of less significant SNPs that are discarded from the logistic regression model.

Chen et al. (2016) showcased a successful application of a deep neural network, D-GEX, to predict expression of target genes based on landmark gene expressions. When compared to traditional linear regression models, D-GEX was able to achieve a relative improvement over the linear model in mean absolute error across two different test datasets of gene expression profiles consisting of microarray and RNA-Seq data.

Neural networks has also seen success in the field of gene analysis, particularly in the case of *convolutional neural networks* (CNN) and *recurrent neural networks* (RNN). CNNs are a class of neural networks inspired by the human visual cortex, and are designed to take into account spatial relationships between input variables. In a convolutional neural network, each perceptron is instead replaced with a $k \times k$ or a $k \times 1$ kernel for 2-dimensional (image) and 1-dimensional (sequential) data, respectively. For an entry of a 1D sequential data with $M \times 1$ vector of features where $M > k$, the convolutional filter is then applied to the first $k \times 1$ sequence of inputs for sequential data, and computes the sum of the element-wise product between the entries in the original data and the kernel, resulting in a scalar output. The filter then convolves (or slides) over 1 element (stride = 1) and computes the output of the next set of $k \times 1$ features until all features have been put through the filter. After the convolutional layer, batchnorm and the activation function can be applied as normal, and the output of the current layer becomes the input for the next layer. The case for an $M_1 \times M_2$ 2D image is similar, where instead of a $k \times 1$ filter, $k \times k$ filters are used to scan over every $k \times k$ sub-matrix of the 2D image to compute the output. By their design, CNNs are widely used in the area of *computer vision*. As a result, CNNs have largely been used in medical classification and segmentation (Ciresan et al., 2012; Li et al., 2014; Esteva et al., 2017). In addition to CNN's successes in computer vision tasks, applications of CNNs are not limited to image data. The ability for the network to detect spatial relationships between data features has made CNNs popular in applications to text and sequential data as demonstrated in Bhandare et al. (2016).

RNNs are a class of neural networks that are designed to model data with a sequential or temporal relationship between the covariates. In a RNN, the network holds the memory of the model state at previous time steps, and the previous state(s) of the model is used to predict the next output along the sequence. Due to the fact that recurrent layers use previous states to make predictions, and that each state is represented by a layer in the network, the network can get very deep when considering all the layers in previous timesteps, which makes the network susceptible to vanishing/exploding gradients. To overcome this issue, Hochreiter and Schmidhuber (1997) developed *long short term memory* (LSTM) units which are capable of handling gradient calculations with long-term dependencies. Chung et al. (2014) introduced *gated recurrent units* (GRU), another type of recurrent unit which serve a similar function to LSTM networks. Other breakthrough research in RNNs has introduced bidirectional RNNs (Schuster and Paliwal, 1997), which are a type of RNN that is trained on both a forward sequence and backward sequence of the input data. The bidirectional architecture has also been extended to LSTM networks (Graves and Schmidhuber, 2005) and GRU networks (Bahdanau et al., 2014). RNNs are currently the state of the art model in the field of *natural language processing*, and have been successfully applied to many types of sequential prediction problems.

Due to the fact that both CNN and RNNs are well suited for handling data with a spatial/sequential relationship between the input features, many studies have applied both RNNs and CNNs to genetic data. Alipanahi et al. (2015) applied a CNN to predict the sequence specificity of DNA and RNA-binding proteins. The resulting model, DeepBind, was able to outperform current sequence specificity prediction models across multiple datasets and evaluation metrics. Another recent application of CNNs in genetics is using the genomic sequence to predict the effects of non-coding genomic variations. Zhou and Troyanskaya (2015) applied a three-layer CNN to predict the effects of the non-coding variations on the chromatin. Zeng et al. (2016) was able to further improve upon the results obtained in Alipanahi et al. (2015) by experimenting with the network architecture. The paper concluded that proper network architecture and model tuning is crucial to the success of the CNNs, and that while a properly designed and tuned CNN was able to outperform DeepBind in prediction tasks, an inappropriately designed network would perform worse than conventional methods.

Hill et al. (2017) used a RNN to predict protein-coding potential of ribonucleic acid (RNA) transcripts. RNNs were chosen in this study due to demonstrated success in learning relationships between input covariates in sequential data, as well as its ability to handle data with inputs of variable length. In this experiment, multiple RNNs were trained in human messenger RNA (mRNA) as well as long non-coding RNA (lncRNA). The best performing RNN according to a validation dataset was able to achieve improved performance when used to predict protein-coding potential, while an ensemble of the five best performing RNNs was able to significantly improve over that result. The RNN models were also able to achieve better generalizability, outperforming previous models when applied to mice transcriptomes.

Given the popularity and demonstrated success of neural networks in genomics, some research has started to look at applying deep learning to a variety of genomic research problems. Cao et al. (2018) and Yue and Wang (2018) both offer an in-depth review of some of the most well-known recent applications of deep neural networks in the field of genomics.

Chapter 4

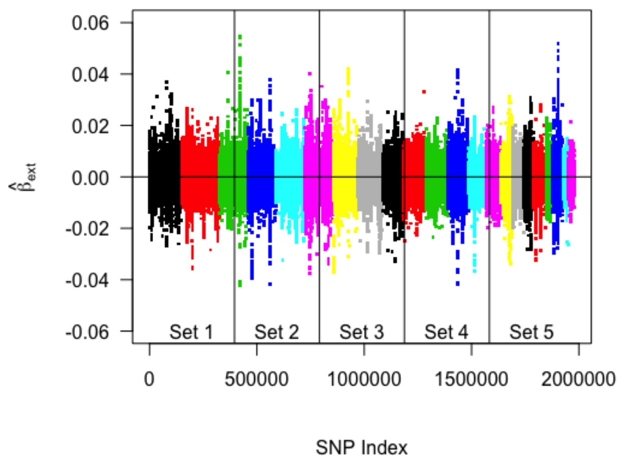
Data and Experiments

4.1 Data

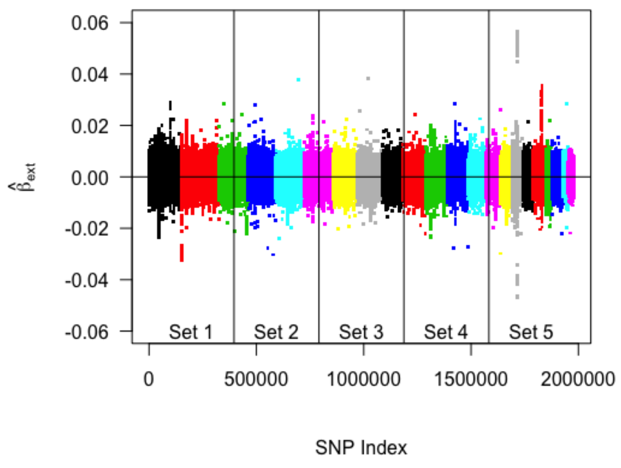
GWAS data from the GIANT consortium (Locke et al., 2015; Wood et al., 2014) was used to calculate $\hat{\beta}_{ext}$ for both height and BMI. To ensure that the comparison between methods was fair, we used the exact same calibration and validation datasets used by Paré et al. (2017), detailed in Section 2.5, with no further pre-processing of the data.

Figure 4.1 depicts a plot of the $\hat{\beta}_{ext}$ ordered by SNP location. We can see the vast majority of points are in a small range around 0, while some points are larger in magnitude, indicating that they were found to be more strongly associated (positively or negatively) with the measured trait in the external GWAS Additionally, we can see that for both measured traits, there are certain loci in chromosomes that were determined by the GWAS to be more associated with the trait, such as in chromosome 3 for height, and in chromosome 16 for BMI.

In the model fitting stage, the 1.98M SNPs were ordered by location on the human genome. The SNPs were then divided into 5 contiguous parts. There were no overlap between the SNPs in the 5-fold split, as to ensure that no SNP was used to predict itself. The chromosomes contained in each split are as follows: set 1 contained chromosomes 1, 2, and part of 3. Set 2 contained the rest of chromosome 3, chromosomes 4, 5, and part of 6. Set 3 contained part of chromosome 6, chromosomes 7 through 9, and part of chromosome 10. Set 4 contained the



(a) Height



(b) BMI

Figure 4.1: Plot of the $\hat{\beta}_{ext}$ for height and BMI ordered by SNP location. The vertical lines represent the 5 splits of the data.

rest of chromosome 10, chromosomes 11 through 13, and part of chromosome 14. Lastly, set 5 contained the rest of chromosome 14, and chromosomes 15 through 22. The dependent variable d (representing amount of deviation between $\hat{\beta}_{ext}$ and $\hat{\beta}_{obs}$) was calculated as:

$$d = (\hat{\beta}_{obs} - \hat{\beta}_{ext}) \times \text{sign}(\hat{\beta}_{ext}) \qquad .$$

The neural network was then trained on 4 of the 5 sets of data, using the absolute value of the regression coefficients, $|\hat{\beta}_{ext}|$, as the univariate input feature. After the model was trained, it was then used to predict the remaining out of sample set to obtain the predicted adjustment factor for the prediction set, $\hat{d}$. This process was repeated 5 times to predict $\hat{d}$ for each set within the split, using the other 4/5th of the data to train the neural network. The resulting constructed weights for the resulting PRS were calculated as:

$$\hat{w} = (|\hat{\beta}_{ext}| + \hat{d}) \times \text{sign}(\hat{\beta}_{ext})$$

Next, LD adjustment was performed on the $\hat{w}$ following the method described in Paré et al. (2017) to obtain the LD adjusted $\tilde{w}_j$. The final PRS was then computed to be:

$$S_i = \sum_{j=1}^{M} x_{i,j}\tilde{w}_j$$

The predictive power of the PRS was measured by the R^2 value between the value of the final PRS and the normalized phenotype values in the validation population. In addition to using the predicted weights, the computed PRS was adjusted for the first 15 *principal components* to account for *population stratification* due to different ancestries. Population stratification refers to the difference in minor allele frequencies present in different populations. According to Price et al. (2006), incorporating principal components into a GWAS as covariates allowed the model to incorporate differences in in minor allele frequencies due to population stratification. To compute the principal components, principal component analysis was performed on the genotype data of the test dataset, and the principal components are ranked in terms of explained variance in the data such all pricnipal components are orthogonal to eachother. To calculate the predicted R^2 of the PRS, a linear model was fitted to the target phenotype, using the PRS and the top 15 principal components as covariates. The prediction R^2 was calculated as:

$$R^2 = \frac{\mathrm{Cov}(g(S), y)^2}{\mathrm{Var}(g(S)), \mathrm{Var}(y)}$$

Where $g(S)$ is predicted value of the measured phenotype from the fitted linear model, and y is the true value of phenotype measured.

4.2 Model Fitting

The neural networks were fitted using the R package *Keras* (Chollet et al., 2015) using *TensorFlow* (Abadi et al., 2016) as the backend package to build the models. The neural network architecture uses $|\hat{\beta}_{ext}|$ as the univariate input feature and d as the target value for regression, as outlined in Paré et al. (2017).

Taking inspiration from the NN architecture in Guo and Berkhahn (2016) and De Brébisson et al. (2015), three separate NNs with 1, 2, and 3 hidden layers were trained with $\{512\}$, $\{1024, 512\}$, and $\{1024, 512, 256\}$ hidden units, respectively, with ReLu activation and a batchnorm layer following the activations. To initialize the weights in the neural network, the weights between the hidden layers were initialized using the Kaiming initialization method. To ensure that the neural network did not overfit to the training data, the input data of the 4 of the 5 splits was further split into a training and validation dataset to monitor the model performance on the validation dataset during the model fitting stage, using 80% of the input data, and using the remaining 20% of data for model validation. The experiments used batchsizes of 256 samples to train the neural networks.

In our experiments, we restricted the number of hyperparameters changed at each iteration of model fitting in order to study the effects each hyperparameter had on the overall predictive power of the NN adjusted risk score. The model was trained with the mean squared error (MSE) loss function, and we used the adam optimizer initially due to its higher reported convergence rate. We experimented with multiple learning rates ranging from 0.1 to 0.001 with a learning rate schedule that decreased the learning rate by a factor of between 0.1 every time the validation MSE did not decrease over the last 5 iterations of training. The models in the initial experimental phase used a dropout probability of 0. The NN was trained for 50 epochs, and the training data was shuffled after each epoch.

In our initial experiments, we encountered issues with exploding gradients for the 2-layer and 3-layer NNs. To remedy this issue, we added gradient clipping to the 2-layer and 3-layer NNs to ensure model stability. We experimented with a range of gradient clipping values from 1 to 100, but found that the neural network performance did not change with the changing clipping values. For the 1-layer NN, clipping value was not needed to prevent exploding gradients.

Model Architecture	Height	BMI
1-Layer	0.0466	0.0195
2-Layer	0.0175	0.0014
3-Layer	0.0174	0.0038

Table 4.1: R^2 Values of PRS as a result of using different NN architectures. The addition of hidden layers severely decreased the predictive power of the neural network.

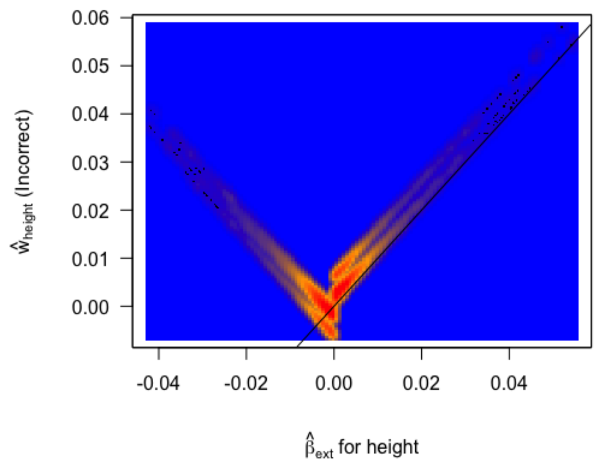
As depicted in Table 4.1, in our initial experiments, we found that the model with the best prediction R^2 was the 1-layer NN with R^2 values of 0.0466 and 0.0195 for height and BMI, respectively. By contrast, the deeper models performed significantly worse than the 1-layer network. The 3-layer NN had the lowest R^2 value out of the three model architectures for height (0.0174), and the 2-layer NN had the lowest R^2 value for BMI (0.0014). The additional of hidden layers did increase the predictive power of the resulting PRS, but quite the opposite. Additionally, the addition of hidden layers to the model resulted in an increase to the model complexity and computational cost. Due to the abundance of downsides associated with using additional hidden layers and lack of improved predictive power, we decided to focus our experiments on 1-layer neural networks only. We observed that the models performed the best with a lower learning rate. Over the course of our tests, we observed that the models had the best results with a learning rate of 0.003.

4.3 Error Report

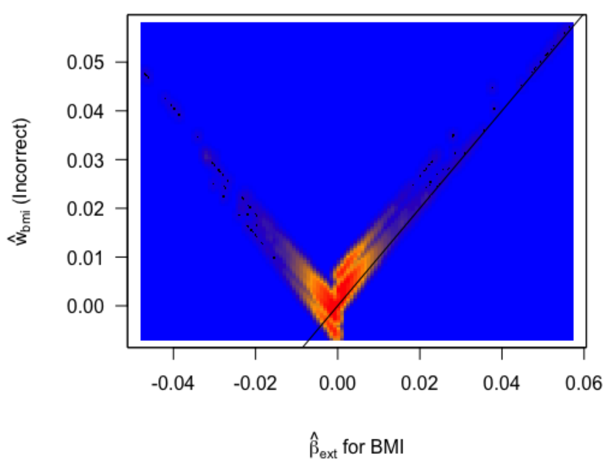
In our initial experiments, we found an error in the formula of the weights generated for the polygenic risk score ($\hat{w}$). In Paré et al. (2017), the formula for $\hat{w}$ (Equation 4) was computed as:

$$\hat{w} = (\hat{\beta}_{ext} - \hat{d}) \times \text{sign}(\hat{\beta}_{ext})$$

We found that most of the predicted adjustment factors, $\hat{d}$ are negative values with a smaller magnitude than $\hat{\beta}_{ext}$ such that for a giving SNP, $|\hat{d}| < |\hat{\beta}_{ext}|$. Under the formula for $\hat{w}$ proposed



(a) $\hat{\beta}_{ext}$ vs $\hat{w}$ for Height



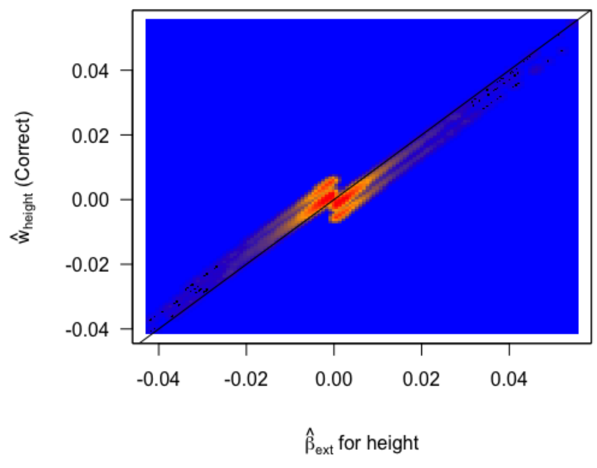
(b) $\hat{\beta}_{ext}$ vs $\hat{w}$ for BMI

Figure 4.2: Plot of $\hat{w}$ against $\hat{\beta}_{ext}$ for both height and BMI under the incorrect formula and the 1-layer NN. The black line depicts the line $y = x$.

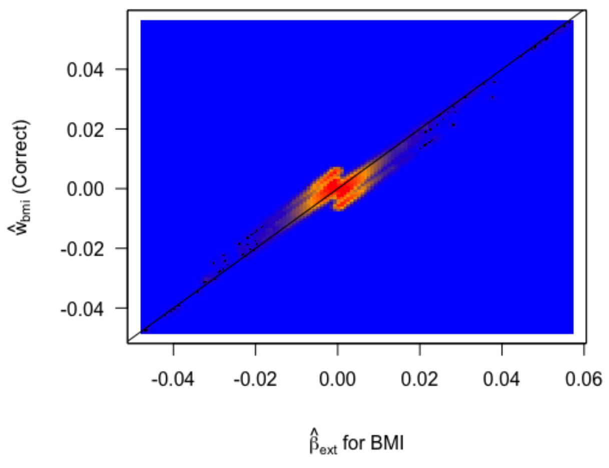
in Paré et al. (2017), when $\hat{\beta}_{ext}$ is negative and $|\hat{d}| < |\hat{\beta}_{ext}|$, $\hat{\beta}_{ext} - \hat{d}$ resulted in a negative number, which when multiplied by sign($\hat{\beta}_{ext}$), resulted in a positive value. When we used this formula, most of the weights in the PRS that should be negative were being adjusted to be positive values.

The effects of the original formula can be seen in Figure 4.2. We observe that under the original formula for $\hat{w}$, when $\hat{\beta}_{ext}$ is negative, the majority of the predicted weights are computed to be positive values, and the $\hat{w}$ did not depict the same behaviour as $\hat{\beta}_{ext}$. Furthermore, we saw that most of the points fell above the line $y = x$. For values of $\hat{\beta}_{ext}$ that were positive, the majority of the adjusted $\hat{w}$ were larger in magnitude than $\hat{\beta}_{ext}$, which was contradictory to the null hypothesis that there is no difference between the external and observed populations. Under the null hypothesis, the $\hat{w}$ should typically be adjusted to be closer to 0 than the $\hat{\beta}_{ext}$.

To examine this further, we requested the source code used for the examples in Paré et al. (2017), and found a discrepancy between the formula presented in Paré et al. (2017) and the source code. From the source code for the GBRT, the weights for the PRS in the code was



(a) $\hat{\beta}_{ext}$ vs $\hat{w}$ for Height



(b) $\hat{\beta}_{ext}$ vs $\hat{w}$ for BMI

Figure 4.3: Plot of the correct $\hat{w}$ against $\hat{\beta}_{ext}$ for both height and BMI using the 1-layer NN. The black line depicts the line $y = x$.

calculated as:

$$\begin{aligned}\hat{w} &= \hat{d} \times \text{sign}(\hat{\beta}_{ext}) + \hat{\beta}_{ext} \\ &= (|\hat{\beta}_{ext}| + \hat{d}) \times \text{sign}(\hat{\beta}_{ext})\end{aligned}$$

We noted two errors with the equation presented in Paré et al. (2017). Firstly, $\hat{w}$ used $\hat{\beta}_{ext}$ whereas the code uses $|\hat{\beta}_{ext}|$. Secondly, the adjustment factor, $\hat{d}$ should have been added to $|\hat{\beta}_{ext}|$ instead of being subtracted.

From Figure 4.3, we saw that under the correct formula for $\hat{w}$, when $\hat{\beta}_{ext}$ is negative and $|\hat{d}| < |\hat{\beta}_{ext}|$, the resulting weight $\hat{w}$ preserved the sign of the original $\hat{\beta}_{ext}$. Additionally, under the corrected $\hat{w}$, the weights were shifted closer to 0 rather than shifted away from 0 as happened with the incorrect formula, which better reflected the null hypothesis of no association with the target population. For values of $\hat{\beta}_{ext}$ greater than 0, the resulting $\hat{w}$ under the correct formula was observed to be mostly less than $\hat{\beta}_{ext}$ in magnitude, with the majority of the points concentrated in a small range around 0. We also saw that when $\hat{\beta}_{ext}$ was greater than 0, a small portion of the adjusted $\hat{w}$ were < 0. This could be attributed to the predicted

Weights	**Height**	**BMI**
$\hat{\beta}_{ext}$	0.165	0.069
$\hat{w}_{correct}$ (1-Layer)	0.1127	0.0236
$\hat{w}_{correct}$ (2-Layer)	0.0562	0.0062
$\hat{w}_{correct}$ (3-Layer)	0.0576	0.0079

Table 4.2: R^2 Values using the corrected PRS formula under different NN architectures compared to the baseline R^2 using the unadjusted $\hat{\beta}_{ext}$.

adjustment factor, $\hat{d}$, having a greater magnitude than $\hat{\beta}_{ext}$ but in the opposite sign as the $\hat{\beta}_{ext}$. In this case, the predicted $\hat{d}$ was < 0 and $|\hat{d}| > |\hat{\beta}_{ext}|$, which shifted $\hat{w}$ to be below 0. This only happened when $\hat{\beta}_{ext}$ was quite close to 0, so there is likely no association between the SNP and the phenotype. Likewise, for values of $\hat{\beta}_{ext}$ less than 0, we observed that the majority of the predicted $\hat{w}$ to were greater than $\hat{\beta}_{ext}$ such that $\hat{w} > \hat{\beta}_{ext}$, with the majority of points concentrated in a range around 0. In the case of $\hat{\beta}_{ext} < 0$, we observed points of $\hat{d} < 0$ and $|\hat{d}| > |\hat{\beta}_{ext}|$, which resulted in a positive value for $\hat{w}$ after multiplying by sign($\hat{\beta}_{ext}$). The shift of $\hat{w}$ towards the null indicated that the majority of predicted $\hat{d}$ appear to be < 0, and that $|\hat{d}| < |\hat{\beta}_{ext}|$, thus preserved the original sign of the $\hat{\beta}_{ext}$. In some cases however, we observed that $|\hat{d}| > |\hat{\beta}_{ext}|$, and resulted in a change in sign from the original $\hat{\beta}_{ext}$.

In addition to the behaviour of the $\hat{w}$ under both formulas, we also observed distinct clusters of $\hat{w}$ depicted in Figures 4.2 and 4.3. The clustering of the $\hat{w}$ appeared to be related to the split of the data, and the behaviour exhibited in the plots will be analyzed in future sections.

We have notified the authors of Paré et al. (2017) regarding the error found in their paper, and they indicated that an erratum will be submitted to correct these errors. As a result of correcting the formula, we also observed an increase in the predictive power of the resulting PRS. Table 4.2 shows the R^2 values of the corrected PRS using the $\hat{d}$ as in Table 4.1. Compared to using the incorrect formula as shown in Table 4.1, we observed that all 3 PRS increased in R^2 when we applied the correct formula. We also observed that similar to the results in Table 4.1, the 1-layer neural network still outperformed the deeper networks in terms of predictive power. We observed that under the incorrect formula, the best performing model was only able to achieve an R^2 of 0.0466 in height and 0.0195 in BMI without any LD adjustment. When we

Number of training epochs	**Height**	**BMI**
50 Training Epochs	0.1127	0.0236
10 Training Epochs	0.1274	0.0552

Table 4.3: R^2 Values of PRS under different training times using the same model architecture.

applied the correct formula for $\hat{w}$ and used the same $\hat{d}$ from the neural network, we were able to get R^2 for height of 0.1127 and 0.0236 for BMI using the 1-layer network. The resulting R^2 values for the PRS showed an improvement in R^2. We also recognized that while the R^2 have improved, they are still below the benchmark result of the unadjusted PRS, which indicated that under the current neural network architecture, the PRS failed to have the same predictive capabilities as the unadjusted $\hat{\beta}_{ext}$. In order to improve the predictive power of the neural network, we further experimented with changing the hyperparameters of the neural network while keeping the 1-layer neural network architecture the same across future experiments.

4.4 Further Model Improvements

From our initial experiments, we observed that the 1-layer neural network with 512 hidden units outperformed deeper models. Additionally, we observed that both the training loss and validation loss stabilized around 10 training epochs for both height (Figure 4.4) and BMI (Figure 4.5), and that further training did not yield any further significant decrease in the training and validation loss. Under the 10 training epochs, we observed a significant increase in R^2 in the same model with the decreased training time. As observed in Table 4.3, for height, there was a 13.04% relative increase in R^2 after decreasing the training epochs, and for BMI, we observed a 133.9% relative increase over the baseline R^2 obtained in our initial experiments. This presented some odd behaviours in our models, as we expected the models to have more predictive power after being trained for longer. However, the results obtained in Table 4.3 were contradictory to our initial hypotheses. Upon further analysis, we concluded that the results obtained in Table 4.3 were due to the high variability present in the predicted $\hat{d}$ and associated $\hat{w}$ rather than any improvement in predictive power in the NN due to shorter training. The high variability in predicted weights will be further discussed in later sections.

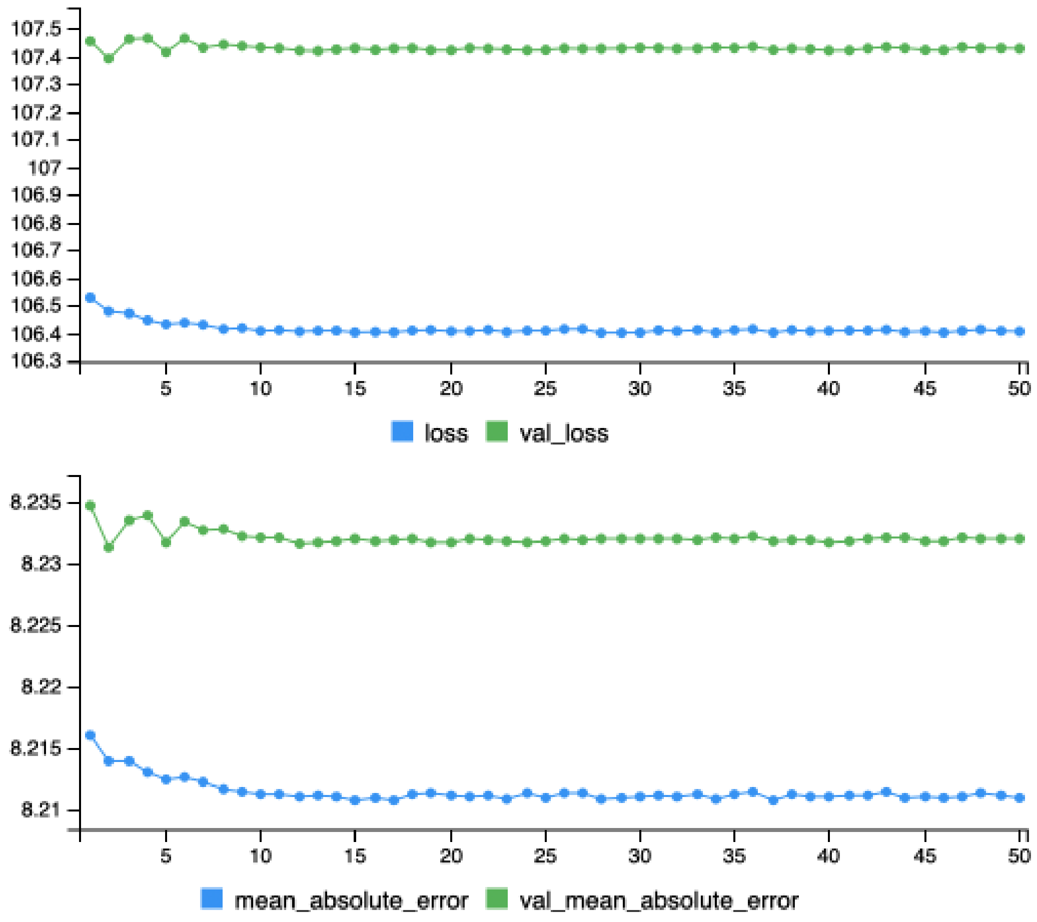


Figure 4.4: Plot of MSE loss function (top) and MAE loss (bottom) for the 1-Layer NN described in the initial model fitting stage over 50 epochs of training. The data is trained on for height to predict the adjustment factor for set 1 of the validation split. The values are scaled up by 1000x for easier visualization. The blue line (loss) represents the MSE loss on the training data, and the green line (val_loss) represents the MSE loss on the validation data

Training epochs	**Height**	**Height (Val)**	**BMI**	**BMI (Val)**
10 Training Epochs	1.0657×10^{-4}	1.0774×10^{-4}	1.0271×10^{-4}	1.0578×10^{-4}
50 Training Epochs	1.0646×10^{-4}	1.0769×10^{-4}	1.0268×10^{-4}	1.0571×10^{-4}

Table 4.4: Mean Squared Error of the 1-Layer NN described in the initial model fitting stage. The NN is trained to predict each set of the 5-fold split, and the MSE is computed on the training and validation (Val) data then averaged across the 5 splits.

While we could not conclude that the reduction in training epochs led to an improvement in predictive power of our neural networks, we did observe that the MSE loss has stopped de-

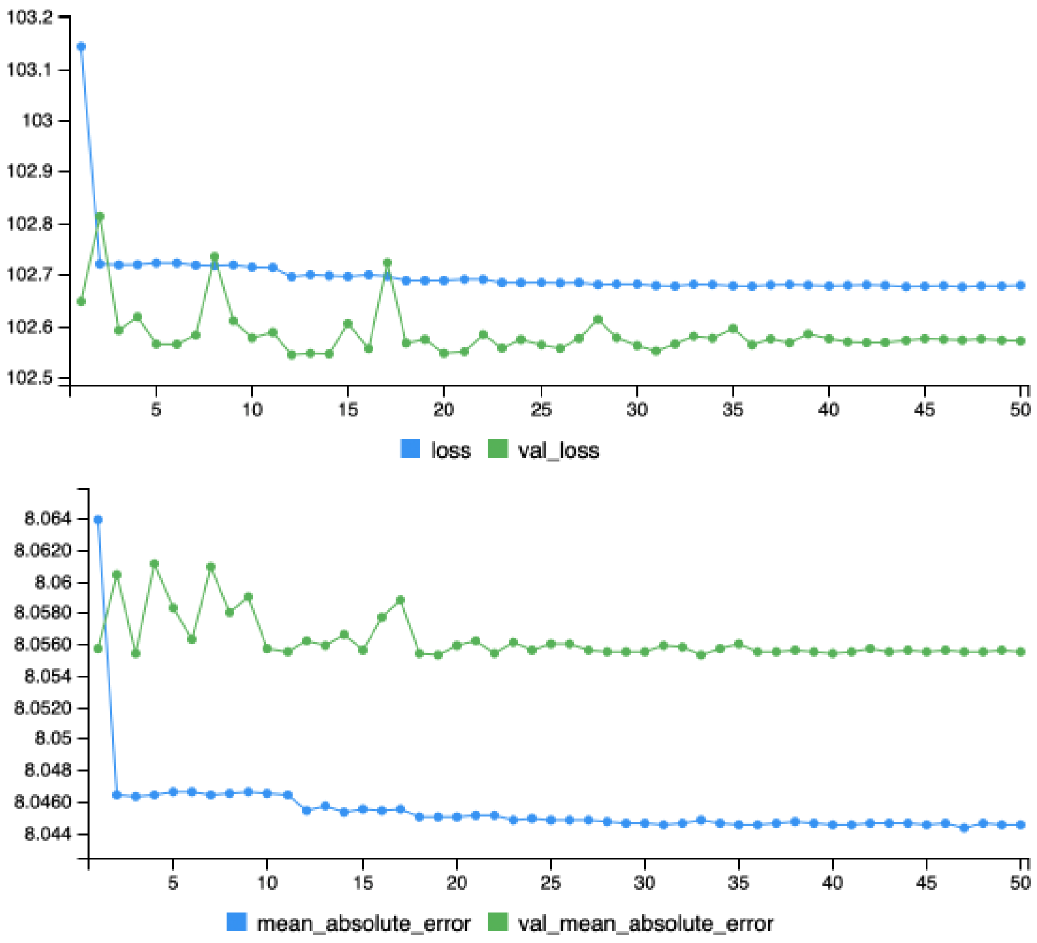


Figure 4.5: Plot of MSE loss function (top) and MAE loss (bottom) for the 1-Layer NN described in the initial model fitting stage over 50 epochs of training. The data is trained on for BMI to predict the adjustment factor for set 1 of the 5-fold split. The values are scaled up by 1000x for easier visualization. The blue line (loss) represents the MSE loss on the training data, and the green line (val_loss) represents the MSE loss on the validation data

creasing after 10 training epochs. We computed the average MSE loss across the model fitting process for all 5 training splits trained on 10 and 50 epochs for both height and BMI, shown in Figure 4.4, and found that increasing the training length to 50 epochs did not result in any significant decrease of MSE in the training or validation data. Thus, we decided to use 10 training epochs to fit our neural networks, as the increased computational efficiency will greatly benefit in rapid fitting of models with no significant drawbacks to the MSE.

Based on the results of comparison between optimizers in Wilson et al. (2017), we replaced

Optimizer	**Height**	**BMI**
Adam	0.1617	0.0572
SGD (Nesterov Momentum)	0.2088	0.0645

Table 4.5: R^2 Values of PRS under different optimizers with the same model architecture. The R^2 values are calculated after performing LD adjustment.

the adam optimizer with SGD using Nesterov Momentum. As can be seen in Table 4.5, the replacement of adam with SGD lead to a substantial increase in R^2, which was congruent with the results found in Wilson et al. (2017). Whereas the model that used the adam performed poorly in its predictive capabilities, switching to SGD lead to a substantial increase in R^2 for both phenotypes after performing the LD adjustment. The NN using SGD saw an increase in R^2 from 0.1617 to 0.2088 for height, beating out the baseline results, and increasing from 0.0572 to 0.0645 for BMI. While the switch to SGD did improve our results, the PRS for BMI still failed to beat the PRS built from $\hat{\beta}_{ext}$ (0.069) even after LD adjustment, and failed to get close to the results using GBRTs achieved in Paré et al. (2017) (0.239 for height and 0.082 for BMI).

We observed that by switching the NN optimizer to SGD, the model was able to generalize better and improve its predictive power. To further improve the out of sample generalization capabilities of our networks, we experimented with different dropout probability of the hidden units between $p = 0$ to $p = 0.5$ to ensure that the model generalizes well across tasks. In our experiments, we found that a higher dropout probability tended to lead to better predicted R^2 results.

In order improve the model's ability to converge to a global minimum, we optimized our models by experimenting with the learning rate and learning rate schedule. We experimented with changing the decay factor of our learning rate scheduler between values of 0.5 to 0.1, in conjunction with changing the initial learning rate. Additionally, we experimented with using a step-wise learning rate schedule, and we did not see any improvement in predictive power with a step-wise learning rate decay.

Weights	**Height**	**BMI**
$\hat{\beta}_{ext}$	0.165	0.069
GBRT	0.169	0.07
Neural Network (Single Model)	0.167	0.065

Table 4.6: R^2 Values of the PRS obtained via different models. The R^2 values are calculated before performing LD adjustment to the weights.

Weights	**Height**	**BMI**
$\hat{\beta}_{ext}$	0.171	0.055
P+T	0.220	0.069
LDPred	0.207	0.074
GBRT	0.239	0.082
Neural Network (Single Model)	0.234	0.074

Table 4.7: R^2 Values of different PRS after LD adjustment was performed on $\hat{\beta}_{ext}$, GBRT, and the NN PRS. The results of using the unadjusted weights ($\hat{\beta}_{ext}$), P+T, and LDPred, and GBRT were obtained from Paré et al. (2017) and the GraBLD package.

Our final NN used the SGD with Nesterov Momentum optimizer with learning rate set to 0.003. The model was trained for 10 epochs, and learning rate decayed by a factor of 0.3 if the validation MSE does not decrease over 5 epochs. The final dropout probability was set to $p = 0.5$. We found that these hyperparameters had the most positive effect on the predictive power of the final PRS. As shown in Table 4.6 Using this network, we were able to achieve a prediction R^2 of 0.167 and 0.065 for height and BMI, respectively, prior to performing the LD adjustment. After performing LD adjustment, as shown in Table 4.7, the resulting R^2 for the neural network were 0.234 and 0.074 for height and BMI. While the models outperformed previously established methods such as P+T and LDPred in R^2 for both height and BMI, it still performed poorly compared to the GBRT results achieved in Paré et al. (2017) (0.239 for height and 0.082 for BMI).

From a comparison of the results in Table 4.6 and 4.7, we observed that just performing LD adjustment on $\hat{\beta}_{ext}$ did not increase the predictive power of the unadjusted $\hat{\beta}_{ext}$ significantly, and resulted in a decrease in R^2 for BMI. Additionally, performing LD adjustment on the unadjusted weights resulted in a less predictive PRS than using P+T. The results indicate that LD adjustment was not an effective method to adjust the PRS by itself. We noted that by combining LD adjustment with a machine learning calibration, we were able to reach a much better R^2 value compared to either methods on their own. Prior to performing LD adjustment, the neural network had an R^2 of 0.167 and 0.065 for height and BMI, similar to that of the unadjusted $\hat{\beta}_{ext}$. After performing LD adjustment, the PRS achieved in increase in R^2 that was much higher than what we saw in the LD adjusted $\hat{\beta}_{ext}$. The same results can be seen with GBRT, where the results of the PRS calibrated from GBRT prior to LD adjustment underperformed relative to P+T in height, and LDPred in height and BMI, performing LD adjustment on the GBRT $\hat{w}$ resulted in a PRS that outperformed both methods for both phenotypes.

4.5 Model Fitting with Additional Data

While optimizing the model hyperparameters had a positive effect on the predictive power of the PRS, we suspected that the predictive power of the NN was limited by only incorporating the $|\hat{\beta}_{ext}|$ as the univariate feature input into the model. We wanted to examine the effect of incorporating other external data as additional features, as we believed that by increasing the dimensionality of the data, we would increase the predictive power of our models. To ensure that the results of models using additional data were easily reproducible, we opted to not incorporate any data that were not widely available from external data sources. We added extra features in two steps.

In the first stage of using additional features, we incorporated additional information available to us from the GIANT consortium, as that was where the external regression coefficients were obtained. In stage 1, in addition to using $\hat{\beta}_{ext}$, we also included the standard error of the $\hat{\beta}_{ext}$, the $-\log_{10}$(p-val) of the regression coefficients, and estimate of the *minor allele frequency* as features. The minor allele frequency (MAF) for a given SNP is defined as the frequency which the minor allele appears in the population. For a given SNP j, the estimate of MAF from the sample is calculated as $\frac{\sum_{i=1}^{N} x_{i,j}}{2N}$. To handle any instances of missing data in the MAF estimates, we replaced any instance of missing data with the mean of the MAF estimates.

Weights	**Height**	**BMI**
NN (Extra Features - Stage 1)	0.167	0.063
NN (Extra Features - Stage 2)	0.144	0.053
NN (Extra Features - Stage 1) - LD adjusted	0.239	0.069
NN (Extra Features - Stage 2) - LD adjusted	0.157	0.072

Table 4.8: R^2 Values of different PRS calibrated using additional input features in the neural network. The Stage 1 extra features incorporates standard error of the $\hat{\beta}_{ext}$, $-\log_{10}$(p-val) of the regression coefficients, and estimate of the MAF as features. The Stage 2 extra features uses (in addition to stage 1 features) the CADD Score, LD Score, and Regulome Score.

In the second stage of model fitting with additional data, we further increased the dimensionality of the data by adding in annotation data including the *Combined Annotation Dependent Depletion* (CADD) score, LD score, and regulome score. The CADD score measures the deleterious (or harmfulness) of a SNP, LD score was derived from Bulik-Sullivan et al. (2015), and is used to compute an estimate of the total LD for each SNP in the genotype data, and regulome score measures the regulatory potential of a SNP. We chose to include annotation data in the second stage of adding additional features because they are widely available for the SNPs, and thus can be easily included in the models. However, because they were not specific to the phenotype we observe or to the dataset used to calculate the external beta values, we did not expect the additional features to have as much predictive power as the additional summary statistics done on the traits observed, hence why we did not include them in the first stage of adding additional features.

To evaluate the predictive power of the extra features, the neural network hyperparameters were kept the same as before. In order to fully optimize the predictive power of the NN using additional features, additional changes to the model architecture and hyperparameters may be needed for a more comprehensive understanding of the predictive capabilities of the extra features.

Table 4.8 summarizes the results of the neural networks trained using additional features. Prior performing LD adjustment, the NN with Stage 1 features showed no improvement in predictive power for height over the univariate model, obtaining an R^2 of 0.169, however, the stage 1 NN also exhibited a slight decrease in R^2 for BMI, with a predicted R^2 of 0.063 compared to the univariate model's R^2 of 0.065. The model with Stage 2 features performs significantly worse across both tasks, obtaining R^2 of 0.144 and 0.053 in height and BMI, results that are below the baseline results from unadjusted weights (0.165 and 0.063).

After performing LD adjustment, we saw an increase in predicted R^2 for both the Stage 1 and Stage 2 models. The Stage 1 model had an increase in R^2 to 0.239 for height and 0.069 for BMI. The R^2 for height in the Stage 1 model managed to outperform both the univariate NN model (0.234) and the GBRT (0.239). Despite its improvement in R^2 for height, the Stage 1 model performed worse for BMI with a predicted R^2 of 0.069 compared to the univariate model (0.074) and GBRT (0.082). Out of all the models, the Stage 2 NN performed the worst in height with an R^2 value of 0.157, below the baseline values. Interestingly, the model also performed marginally better than the Stage 1 model in predicted R^2 for BMI with a value of 0.072. Overall, the Stage 2 model only managed to achieve a decrease in predictive power for the NN compared to just using $|\hat{\beta}_{ext}|$, while the Stage 1 model had a 0.84% relative increase in R^2 for height, but also had a 6.76% relative drop in R^2 for BMI over the univariate model.

4.5.1 Analysis of Increased Variance in Neural Networks

We observed that including extra summary statistics as features slightly improved the R^2 for height, but resulted in decreased the R^2 for BMI, whereas including the additional annotation data as features resulted in a decrease of R^2 in both traits. Upon further analysis through running the experiment multiple times (with only Stage 1 features), as shown in Figure 4.6, we observed that there was an increase in the variability of predicted weights that were higher in magnitude, which were the values that were most influential in the PRS. Figure 4.6 showed that across multiple runs of the stage 1 NN, the standard deviation increased to 0.08 for height and 0.4 for BMI at higher values of $\hat{w}$. We also noted that the standard deviation between the $\hat{w}$ across multiple runs was in in some cases much higher than the $\hat{w}$ itself, and was particularly prevalent in the more extreme weights. This presented a problem, as due to the high variability of the $\hat{w}$ across multiple runs, the weights in the resulting PRS could have different signs across different experiments, and could have drastic effects on the overall predictive power of the PRS.

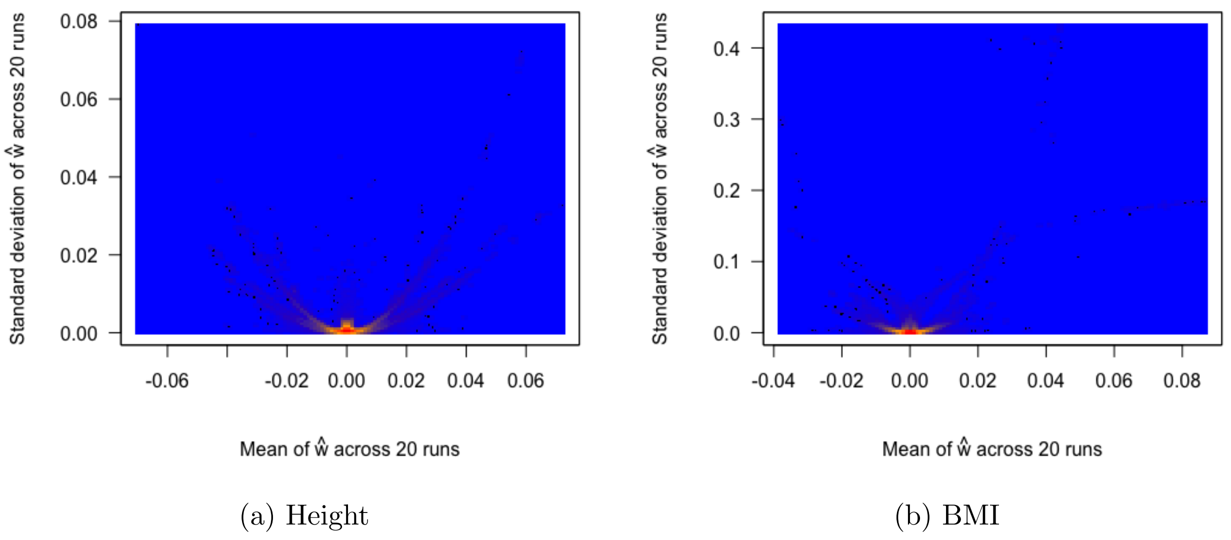


Figure 4.6: Plot of the mean and standard deviation of $\hat{w}$ across 20 experiments using additional summary statistics as input features.

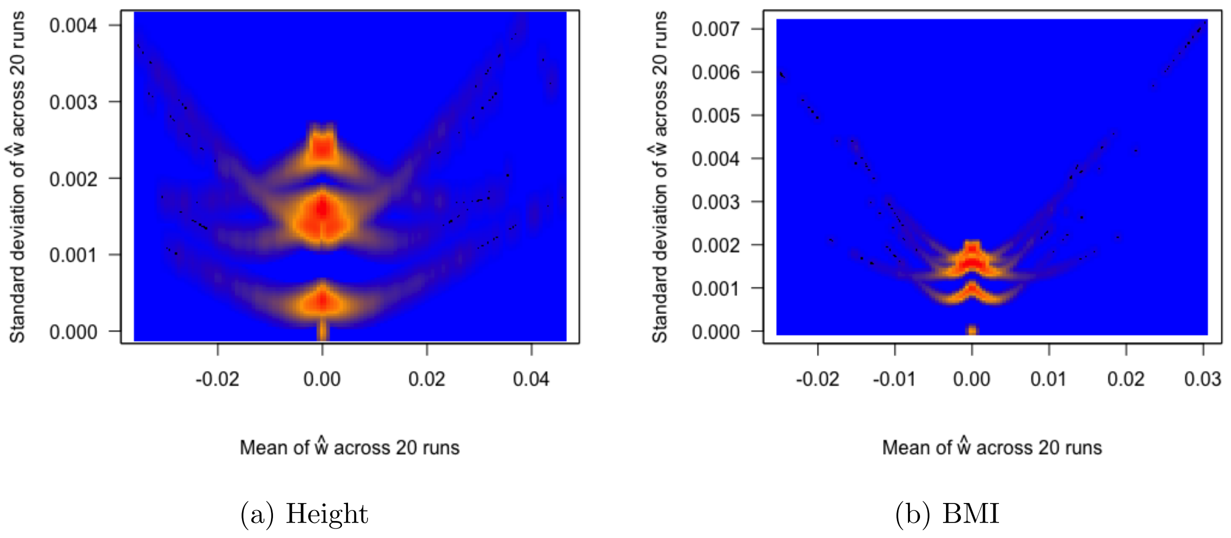


Figure 4.7: Plot of the mean and standard deviation of $\hat{w}$ across 20 experiments using only $|\hat{\beta}_{ext}|$ as the univariate input feature.

When we used just the univariate features, we did not observe the same degree of variability in the extreme weights predicted by the neural network, as can be seen in Figure 4.7. Additionally, unlike the NN with additional features, we did not see any instances where the standard deviation was much higher than the mean value of $\hat{w}$ across the multiple experiments, which indicated that there were no instances of the more extreme weights switching signs across the experiments. We observed that while the standard deviation did increase for more extreme weights in some cases, that the standard deviations of the predictions with univariate features were much less than with the additional features. This suggested that the addition of extra features increased the variability of the predicted $\hat{w}$, leading to a much more variable PRS. The variance in the extreme weights could explain the drastic difference in R^2 values obtained the neural network with additional features. Due to the instability of the predictions, we do not recommend the weights generated from the multivariate models as a preferred PRS, and instead focused our experiments on using neural networks trained on the univariate regression coefficients $|\hat{\beta}_{ext}|$.

In addition to the increased variability of the predicted $\hat{w}$ at the more extreme weights, we also observed several odd behaviours in our plots. In Figure 4.7 we observed clusters of points with 0 mean and 0 standard deviation in both height and BMI, which indicated that there were points in the data where the models were consistently calibrated to have $\hat{w} = 0$. Additionally, Figure 4.7 depicted distinct clusters of points consistent with what was shown in Figure 4.2, which suggested that the relationship between the mean and standard deviation of $\hat{w}$ across the 20 runs of the experiment appeared to be dependent on the split of the data. Lastly, we also observed from Figures 4.6 and 4.7 that the variability in the predicted $\hat{w}$ across multiple runs was much higher for BMI than for height. These behaviours exhibited in the figures will be further analyzed in the discussions.

4.6 Model Ensembling

One important issue we observed is the variability in the predicted $\hat{w}$, and by extension, the variability of the predictive power of the resulting PRS. Due to the usage of SGD, He weight initialization, dropout, and the shuffling in the training data, we recognized that there will be some variability in the predictive power of the PRS generated by the models. One effective way to reduce the variance in the predicted $\hat{d}$ would be to use *model ensembling.* Model ensembling functions by taking a linear combination of predictions taken from multiple models and uses

Ensembles	**Height**	**BMI**
Ensemble 1	**0.2405**	0.0685
Ensemble 2	0.2404	0.0665
Ensemble 3	0.2361	0.0655
Ensemble 4	0.2373	0.0696
Ensemble 5	0.2368	0.0663
Overall Average	0.2388	**0.0761**

Table 4.9: R^2 Values of different PRS generated by Ensembles after LD adjustment. Each ensemble was created by taking the average predicted $\hat{w}$ of a subset of 5 models (selected randomly without replacement), and the overall average takes the average $\hat{w}$ across the 20 runs. The best results obtained by model ensembles are highlighted in bold.

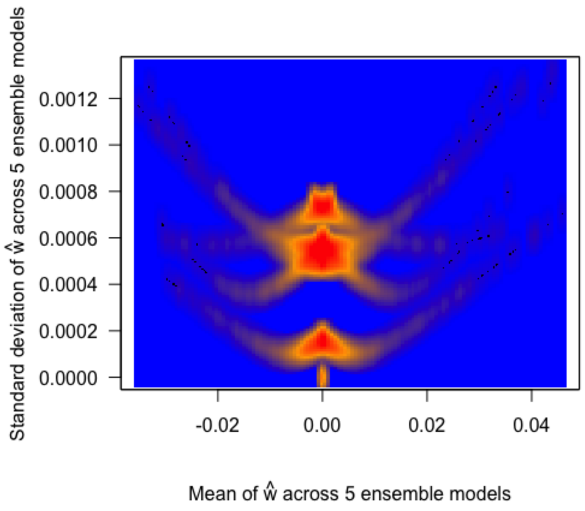
the combined predictions as the final output. In the case of averaging the predictions made by the models, model ensembling can be thought of as a variation of the bagging algorithm, where at each step, instead of taking a bootstrap sample of the data to fit the model, we use the entirety of the training data. Appendix A.6 details how model ensembling may be able to reduce the variability of model predictions.

To test the effect of model stacking on the variability of the predictions, we took subsets of 5 randomly chosen models fitted using the univariate feature and averaged each subset to create 5 ensembles predictions, then plotted the mean and standard deviation of the ensembled predictions. We also used the predicted weights using the average of the 20 predictions used those ensembled values to calculate PRS to test the predictive power of the overall ensemble PRS.

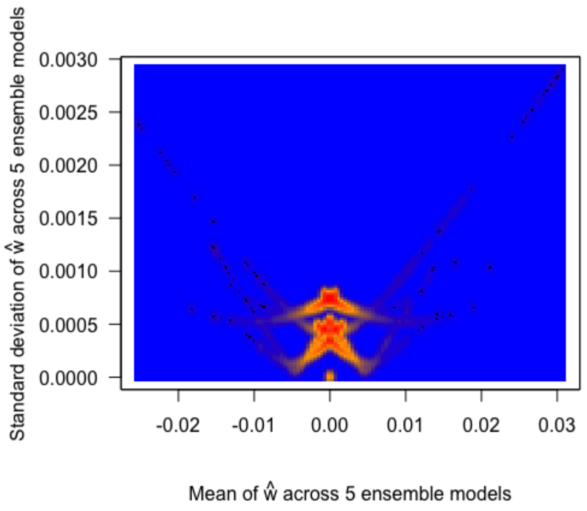
Table 4.9 summarizes the results of using ensembled neural networks. We observed an increase in R^2 across all ensembled models for height when using the univariate feature. The best ensembled model, ensemble 1, had an R^2 of 0.2405 compared to the single NN's R^2 of 0.234. However, with the exception of the overall ensemble model, we saw a slight decrease in prediction R^2 for ensembled models for BMI when compared to the single model. From the ensembles that contained the subsets of the 20 runs, the best performing model achieved an R^2 of 0.0696, lower than the single model's R^2 of 0.074. The lower performance achieved by the ensembled models suggests that perhaps the results achieved in the single model for BMI was an outlier. The only ensemble model that outperformed the single model was the overall model average, achieving an R^2 of 0.0761. We observed that by averaging the predictions of all the single models, we were able to create an ensemble model that outperforms the single model in prediction R^2 for both height and BMI.

4.6.1 Effects of Ensembled Models on Variability of $\hat{w}$

As seen in Figure 4.8, we plotted the means and standard deviations of $\hat{w}$, but this time the $\hat{w}$ comes from ensembled models. When compared with Figure 4.7, we observed a decrease in the standard deviation in the extreme weights in the resulting PRS for both height and BMI. For height, the highest standard deviation between the ensembled weights was approximately 0.0014, down from approximately 0.004 from single model predicted weights. For BMI, the highest standard deviation dropped from approximately 0.007 to 0.003 when the models are ensembled. In both traits, by using ensembled models, we observed a reduction of over one half in standard deviation of $\hat{w}$ across the multiple models. The reduction in standard deviation verified that ensembling is a useful technique to reduce the variability of the computed PRS, thus reducing the variability in the predictive power of the neural networks. This technique for reducing prediction variance can also be extended to the multivariate feature case. However, due to the higher variance associated with the $\hat{w}$ for the multivariate models, more models would be needed in the ensemble in order to obtain a lower level of standard deviation similar to the ensembled univariate models. Additionally, by ensembling models, we observed a slight change in the relationship between the mean and standard deviation of the weights, which we believed to be the result of the randomly chosen subsets. Despite the reduction in standard deviation, we still observed some odd behaviours in Figure 4.8, which were consistent with what we observed in Figure 4.7, such as increase of standard deviation at extreme weights, and cluster of points with zero mean and zero standard deviation.



(a) Height



(b) BMI

Figure 4.8: Plot of mean and standard deviation of $\hat{w}$ across 5 ensembled models.

Weights	**Height**	**BMI**
$\hat{\beta}_{ext}$	0.165	0.069
P+T	0.220	0.069
LDPred	0.207	0.074
GBRT	0.239	0.082
Neural Network (Single Model)	0.234	0.074
NN (Extra Features - Stage 1)	0.239	0.069
NN (Extra Features - Stage 2)	0.157	0.072
Overall NN Ensemble	**0.239**	**0.076**

Table 4.10: R^2 Values of different PRS compared to machine learning adjusted PRS. The R^2 is calculated after performing LD adjustment. The best performing neural network PRS is highlighted in bold. The NN ensemble is able to match the results in predictive power for height achieved in Paré et al. (2017), and only outperformed by GBRT in predictive power for BMI.

4.7 Summary of Results

Table 4.10 summarizes the predictive power of all the methods discussed, measured in prediction R^2. The ensemble containing 20 1-layer neural networks, trained on the univariate feature $\hat{\beta}_{ext}$ obtained better results in terms of prediction R^2 when compared to all single models for both height and BMI, while also having lower variability of predictions compared to single models. For height, the NN ensemble achieved an R^2 of 0.239, outperforming the single NN (0.234), P+T (0.220), LDPred (0.207), and the unadjusted weights (0.165), but did not outperform the results obtained by the GBRT in Paré et al. (2017) (0.239). For BMI, the NN ensemble achieved an R^2 of 0.076 and managed to outperform the single NN (0.074), as well as P+T (0.069), LDPred (0.074), and the unadjusted weights (0.069). The NN came up short when compared to GBRT (0.082) which resulted in a relative decrease of 7.3% in R^2. We observed that the NN was particularly good at predicting height compared to BMI, achieving results comparable to GBRT, but performs worse in BMI.

Despite getting favourable results, we cannot say that we have fully optimized the hyperparameters for the NN, simply due to the vast combinations of hyperparameters that could be tuned. We have yet to implement any form of regularization such as L_1/L_2 regularization, weight decay, or use a cyclical learning rate schedule into our models, thus we do not know the results of changing those hyperparameters on the final PRS. It is possible that by further changing the hyperparameters, we can improve the results in BMI to outperform the GBRT. We also noticed that our current NN models tend to perform very well in predictive power for height, but generally worse than GBRT in BMI, it may be possible that the current NN architecture is better suited for calibrating the PRS for height than for BMI. As a result, changing the hyperparameters may improve the predictive power of the NN calibrated PRS for one trait, but may come at a cost of decreased predictive power for the other trait. It may be possible to use multiple different NN architectures to predict each phenotype separately. While this approach may yield improvements over current results, by specializing models towards a specific phenotype, we would lose the ability for the NN to generalize across multiple tasks.

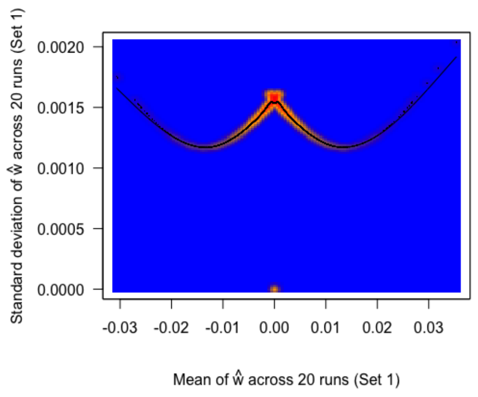
Chapter 5

Further Analysis of Fitted Models

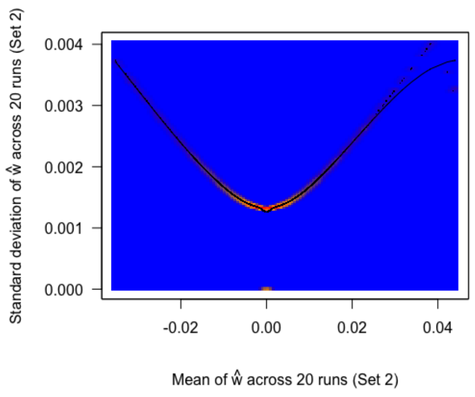
5.1 Distinct Clusters of $\hat{w}$

One observation we made across experiments using both univariate and multivariate features was the multiple distinct clusters of predicted $\hat{d}$'s (as well as the associated $\hat{w}$'s) in Figures 4.6 and 4.7. Further analysis revealed that each of the distinct clusters of predicted points corresponded to one of the 5 prediction sets. Additionally, in in Figures 4.6 and 4.7, we also observed distinct relationships between the mean and variance across the different sets of $\hat{w}$.

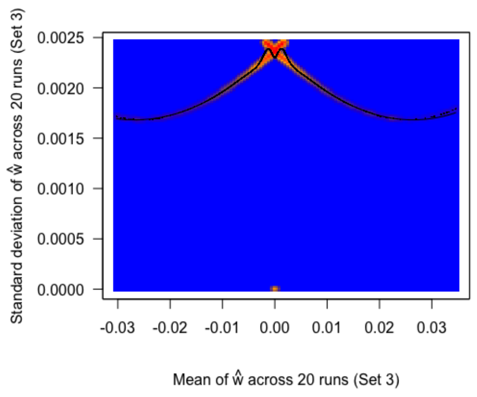
As seen in Figure 5.1, in each of the sub-figures, we plotted the mean of the $\hat{w}$ against the standard deviations of $\hat{w}$ for each of the sets in the 5 splits of the data. As shown in Figures 5.1a and 5.1e, for SNP Sets 1 and 5, we observed that as the magnitude of the weights increased, the standard deviation initially decreased, but then increased again past a certain threshold. Figure 5.1c depicted a similar relationship between the mean and standard deviation, except that the standard deviation continued to decrease as the magnitude of the weights increased, and only showed a slight increase in standard deviation in the tail end of the plots. Figure 5.1c was the only set across all 5 splits of the data that showed a lower standard deviation in the extreme values of $\hat{w}$ compared to the standard deviation of $\hat{w}$ near 0. For SNP Sets 2 and 4, seen in 5.1b and 5.1d, the standard deviation for the predicted weights in those sets almost strictly increased as the magnitude of the weights increased. For 5.1b, we saw that as the weights became larger, the associated standard deviation showed a concave down pattern,



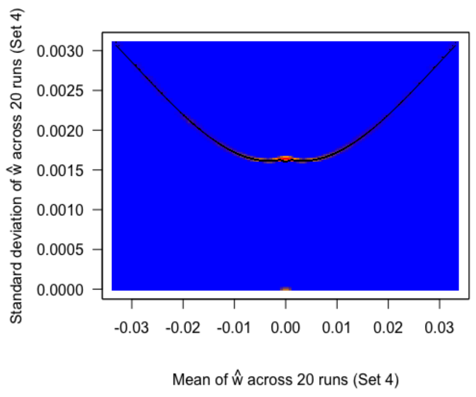
(a) Set 1



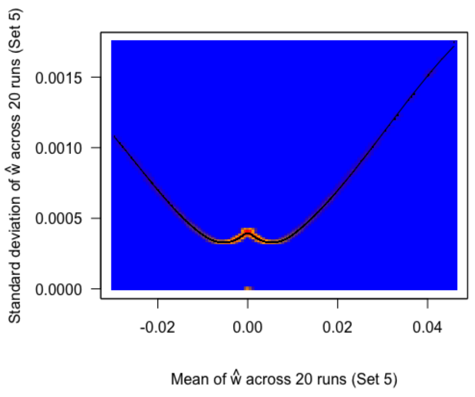
(b) Set 2



(c) Set 3



(d) Set 4



(e) set 5

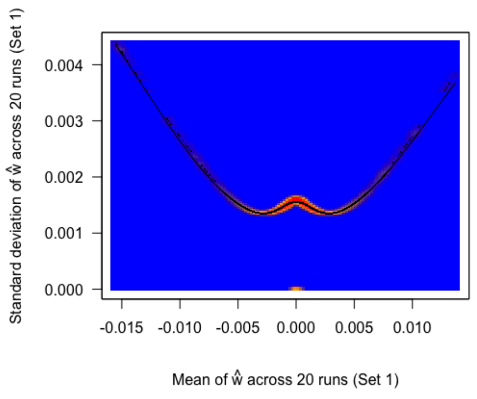
Figure 5.1: Plot of Mean vs Standard Deviation of each set of $\hat{w}$ for height across 20 runs, split up according to the 5 contiguous splits.

indicating that the standard deviation is increasing at a slower rate. One interesting behaviour of note in Figure 5.1 is that in each of the sub-figures, we observed a cluster of points in every plot with a mean and standard deviation of 0.

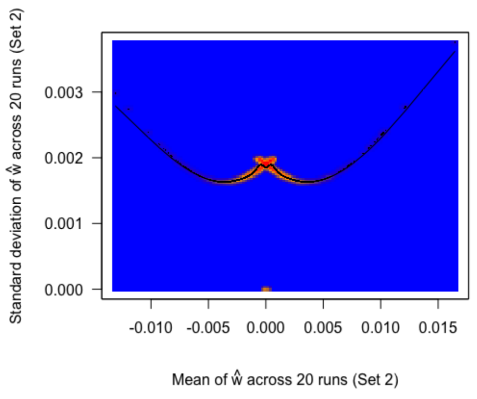
The relationships between the mean and standard deviation for weights in the PRS for BMI as shown in Figure 5.2 displayed a more consistent behaviour across all sets compared to Figure 5.1. We observed that as the mean of the $\hat{w}$ moved farther away from 0, there was a decrease in standard deviation up until a certain point, where the standard deviation then began to increase, which resulted in a higher standard deviation in the extreme values compared to weights that are closer to 0. Similar to 5.1, there was a small cluster of points with mean and standard deviation of 0 in each of the sub-figures, which indicated that across 20 models, there were a small portion of $\hat{w}$ that were consistently being predicted to have a value of 0 with no variation. This was consistent with the observations made in Figure 5.1. This behaviour would be investigated in full detail in a later section.

A box plot of the mean values of $\hat{w}$ in the 5 splits for both phenotypes is in Figure 5.3. For height, it appeared that each separate predicted split had very similar median values to one another, and that the range of values also appeared to be quite close. However, for BMI, we saw that the predicted weights for SNP Set 5 had a wider range of distributions in the predicted weight compared to the rest of the sets. The more extreme values of $\hat{w}$ shown in that set indicated that certain SNPs in the 5th set were being calibrated to be more influential on BMI compared to the rest of the SNPs.

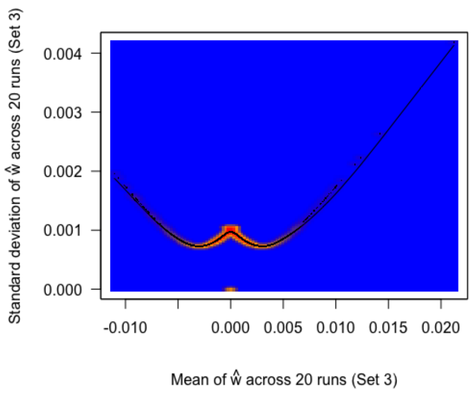
Figure 5.4a showed more drastic differences in standard deviations of weights for each of the 5 splits across the 20 runs of the experiment. In contrast to the distribution of the mean values, the median values for standard deviation for height (Figure 5.4a) were not very close to one another, particularly for set 3 which had a higher median value than all the rest, and set 5 with the lowest median value for standard deviation. Additionally, SNP Set 1 showed a distribution of the standard deviation values that were much closer to the Interquartile Range (IQR) than the rest of the sets, Sets 2, 4, and 5 displayed large amounts of outliers that are



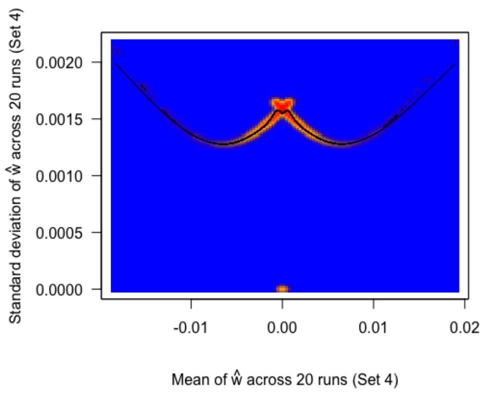
(a) Set 1



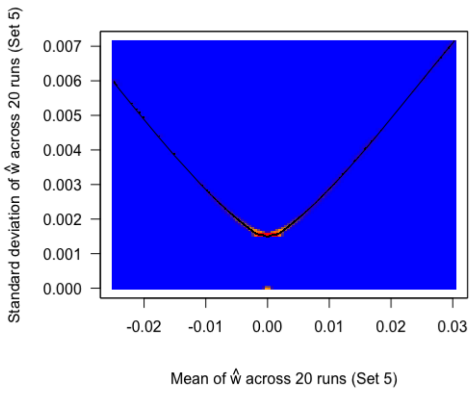
(b) Set 2



(c) Set 3

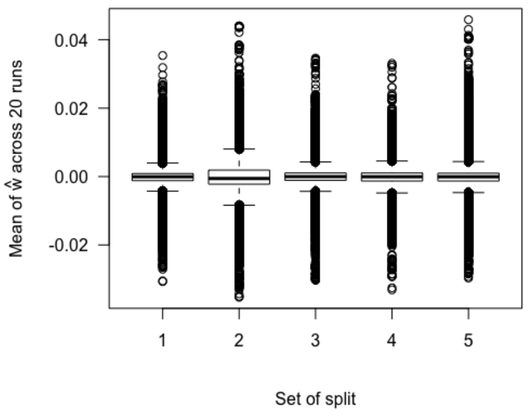


(d) Set 4

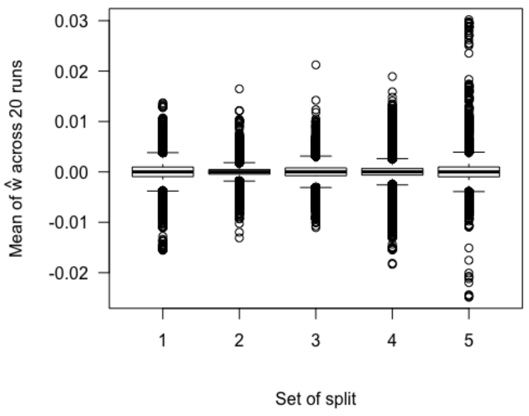


(e) set 5

Figure 5.2: Plot of Mean vs Standard Deviation of each set of $\hat{w}$ for BMI across 20 runs, split up according to the 5 contiguous splits.

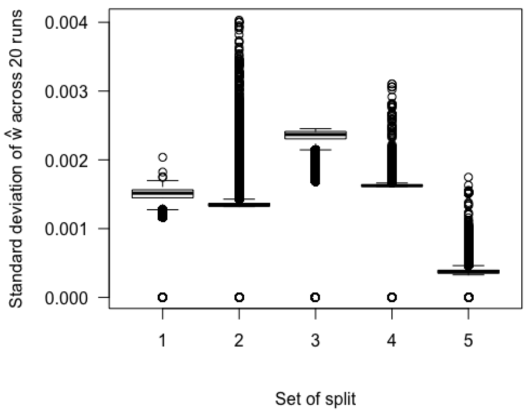


(a) Height

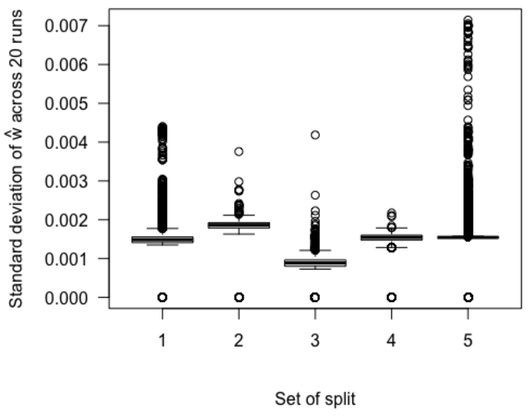


(b) BMI

Figure 5.3: Box plot of each distinct set's mean across 20 runs of the experiment.



(a) Height



(b) BMI

Figure 5.4: Box plot of each distinct set's standard deviation across 20 runs of the experiment.

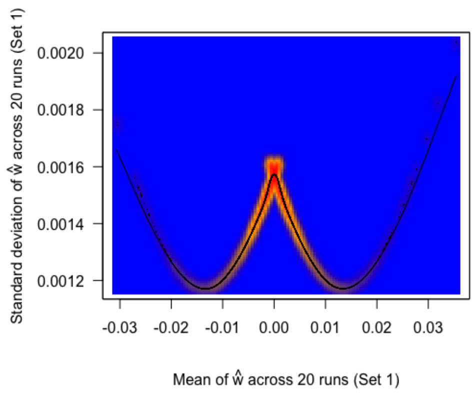
above the third quartile (Q3), but very few values below the first quartile (Q1), which indicated an asymmetrical distribution of the data. Likewise in Figure 5.4b, sets 2 and 3 had different median values compared to sets 1, 4 and 5, but not at the scale shown in Figure 5.4a. Each set also had an asymmetrical distribution of the outliers, with outliers lying above of Q3 at a much higher frequency than lower than Q1. While the range of standard deviation was higher for BMI, the boxplots for BMI appeared to be much more consistent across the 5 splits than for height. This is further reinforced the behaviour of the $\hat{w}$ shown in the plots of each distinct set of predicted weights in Figure 5.1 and 5.2. In Figure 5.2, while the range of standard deviation was wider for BMI, the shape of the pattern displayed by the plot of the mean against the standard deviation remained fairly the same. By contrast, in Figure 5.1, for the behaviour of $\hat{w}$ for height, there was a higher discrepancy in the shape of the pattern displayed by the relationship between the mean and standard deviation, but there was less standard deviation overall between the weights in the PRS for height than there was in BMI.

One other interesting behaviour depicted in the plots is that there appears that in certain plots, most noticably seen in Figures 5.1a and 5.1c for height, as well as 5.2b and 5.2d for BMI, there appeared to be a crossing over of two curves in a small range around 0, which displayed an "X" like pattern. It appeared that the calibrated $\hat{w}$ were separated into two distinct groups based on whether they were positive or negative, and each group had its own relationship between the mean and standard deviation as displayed in the plots. This appears to be consistent with the behaviour of the $\hat{w}$ shown in Figure 4.3, where there appeared to be a separation of values based on whether $\hat{\beta}_{ext}$ is positive or negative. This "X" like pattern was likely due to the switching of the signs in the final $\hat{w}$, which occurred when $|\hat{d}| > |\hat{\beta}_{ext}|$ and values of $\hat{\beta}_{ext}$ were very close to 0, as discussed in Section 4.3. The plots indicated that the relationship between the mean and standard deviation of the resulting weights was related to values of $\hat{\beta}_{ext}$. This showed a promising sign that the neural network was capable of learning a relationship between the input and dependent variables, rather than making predictions that were uncorrelated with the features.

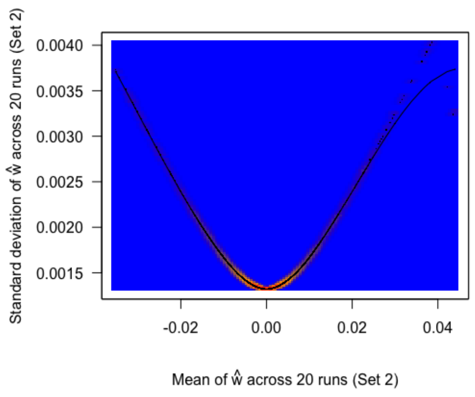
5.2 Effect of $\hat{\beta}_{ext} = 0$ in the PRS

As noted in Figure 5.1 and 5.2, we observed a cluster of points in the plot of every predicted set where across 20 runs, we consistently obtained a predicted $\hat{w}$ of 0. We discovered that this behaviour was due to the presence of $\hat{\beta}_{ext}$ with values of 0 in the input features which always results in a final weight of zero irrespective of the value of $\hat{\beta}_{ext}$ or any other input features to the neural network.

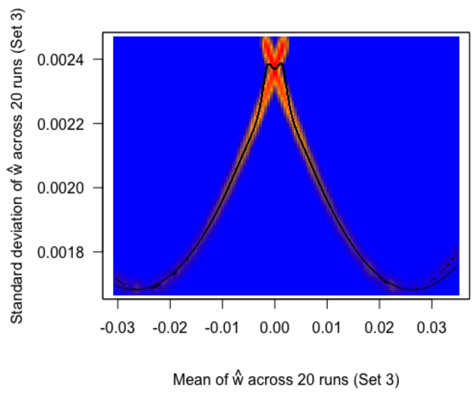
In the external summary statistics obtained from the GIANT consortium, we observed



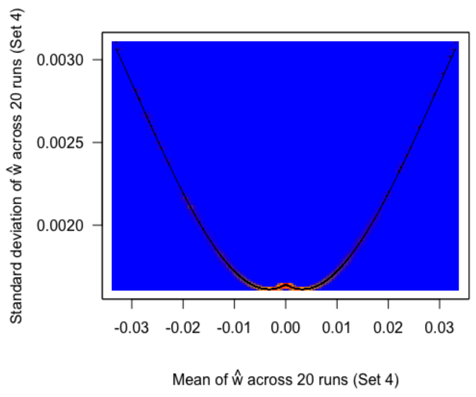
(a) Set 1



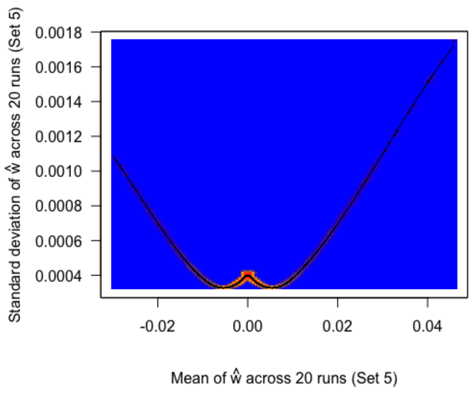
(b) Set 2



(c) Set 3

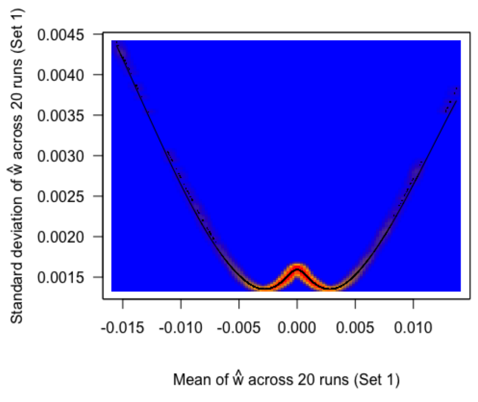


(d) Set 4

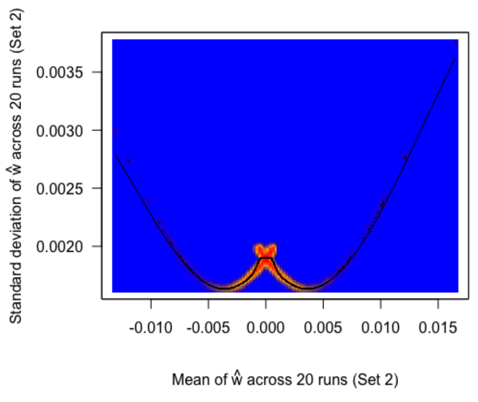


(e) set 5

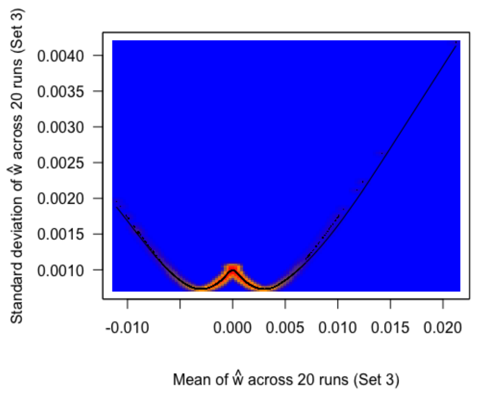
Figure 5.5: Plot of mean vs standard deviation of each set of $\hat{w}$ for height across 20 runs after removing the weights where $\hat{\beta}_{ext} = 0$.



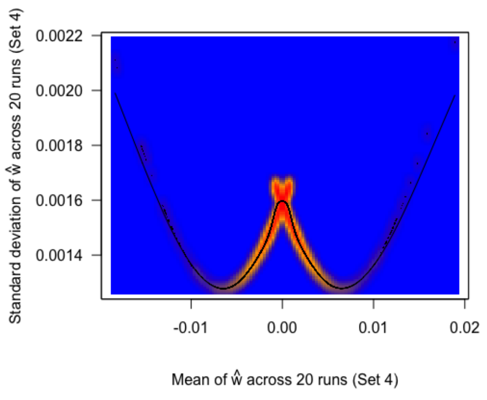
(a) Set 1



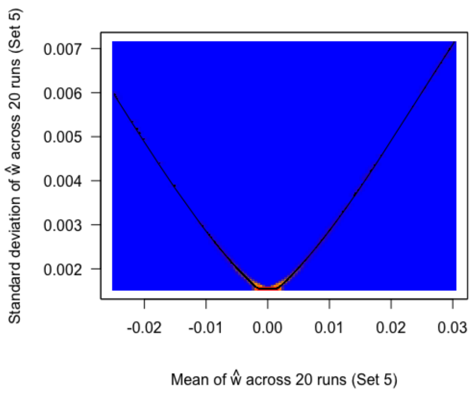
(b) Set 2



(c) Set 3



(d) Set 4



(e) set 5

Figure 5.6: Plot of mean vs standard deviation of each set of $\hat{w}$ for BMI across 20 runs after removing the weights where $\hat{\beta}_{ext} = 0$.

13,939 instances where $\hat{\beta}_{ext} = 0$ for height, and 14,728 instances where $\hat{\beta}_{ext} = 0$ for BMI across 1.98M SNPs. There was an overlap of 274 SNPs where the $\hat{\beta}_{ext}$ was 0 for both height and BMI. By contrast, we only observed 7 instances of SNPs in the calibration sets for height and BMI where $\hat{\beta}_{obs} = 0$, and no instances of any overlaps of SNPs with $\hat{\beta}_{obs} = 0$ across both calibration sets. Lastly, we observed 4 instances in both height and BMI where both the $\hat{\beta}_{ext}$ and $\hat{\beta}_{obs}$ for a SNP is equal to 0. Figures 5.5 and 5.6 depict the same plots as Figures 5.1 and 5.2 after removing the instances where input $\hat{\beta}_{ext}$ had a value of 0. The resulting plots of the mean and standard deviation of the predicted weights no longer exhibit any clusters of 0's, and the behaviour of the standard deviation in relation to the mean became much more consistent across small values. We also examined the standard errors of the $\hat{\beta}_{ext}$ using the external GWAS, and found no instances where the standard error is equal to 0. This ruled out the possibility that the instances of $\hat{\beta}_{ext} = 0$ is due to that particular SNP not being studied in the external GWAS. We further examined the $-\log_{10}(p\text{-val})$ of the $\hat{\beta}_{ext}$ with 0 values and found that for height, the $-\log_{10}(p\text{-val})$ of the $\hat{\beta}_{ext}$ were either 0, 0.004364805, 0.337242168, or 0.585026652, corresponding to p-values of 1, 0.99, 0.46, and 0.26, respectively, with the majority of the SNPs having a p-value of 1. For BMI, the $-\log_{10}(p\text{-val})$ of the 0 $\hat{\beta}_{ext}$ were either 0, or range from 3.619687×10^{-3} to 0.3613111, corresponding to a range of p-values from 0.4352 to 1, again with the majority of p-value being equal to 1. The p-values for all the $\hat{\beta}_{ext}$ were very high, thus indicate that the corresponding SNPs have a statistically insignificant association with the measured phenotype. We believed that due to the high p-values associated with the SNPs, instances of $\hat{\beta}_{ext} = 0$ with an associated p-value of 1 were a result of the external GWAS which concluded that the SNPs had no association with the measured trait, and instances of $\hat{\beta}_{ext} = 0$ and p-value not equal to 1 were values very close to 0 that were set to be 0 due to rounding in the data we had available.

We identified two ways in which a $\hat{\beta}_{ext}$ of 0 could affect the resulting PRS. Firstly, in the model fitting stage, an input $\hat{\beta}_{ext}$ of 0 would have resulted in a 0 output in the forward pass. Since we were updating the weights in the neural network by calculating the gradient of the MSE loss function, and the dependent variable $d = 0$, any instances of $|\hat{\beta}_{ext}| = 0$ would have incurred a loss of 0 in the model training, thus lowering the overall MSE compared to if the

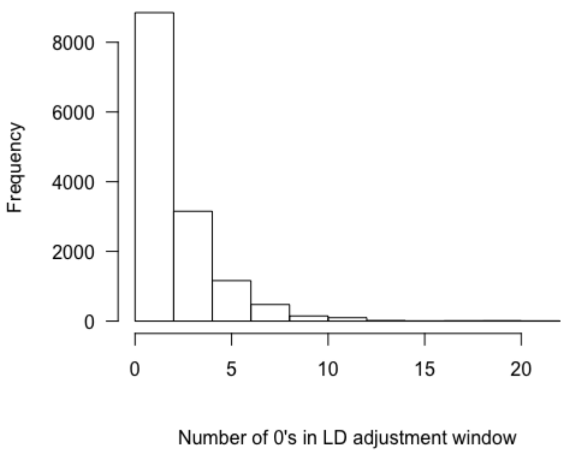
PRS	**Height**	**BMI**
Original NN	0.1668	0.0674
Original NN (LD adjustment)	0.2356	0.071
NN trained without 0's	0.1647	0.064
NN trained without 0's (LD adjustment)	0.2323	0.0684

Table 5.1: R^2 Values of PRS generated by the neural network trained without 0's.

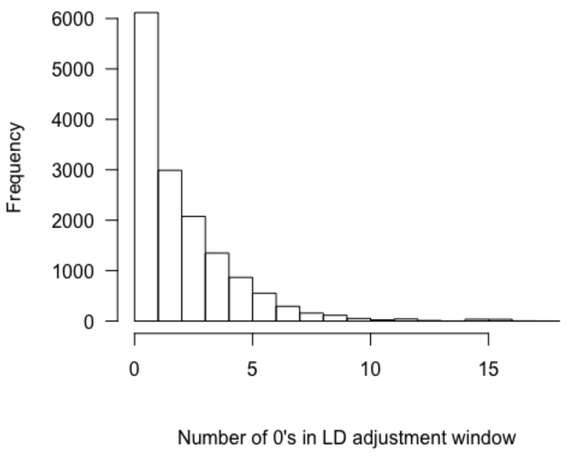
0's were removed from the data, which resulted in a smaller weight adjustment to the network. The second way that the resulting PRS could have been affected was in the LD adjustment stage. While a $\hat{w}$ of 0 would not have affected the PRS prior to performing LD adjustment, we believed that removing these weights from the PRS could potentially have an effect on the predictive power of the PRS post LD adjustment. Due to the process of the LD adjustment, if these weights of 0 had some pairwise correlation with other SNPs in the LD adjustment window, then it would contribute to overall LD adjustment value η, which then would down-weight any weights of SNPs with a non-zero pairwise r^2 in the risk score.

To further examine the impact of the $\hat{\beta}_{ext}$ of 0 on the polygenic risk score during the model fitting stage, we performed experiments by removing the 0's from the data at the model fitting stage. To remove variability between multiple models, we set the same seed across both experiments, then ran 2 experiments using the same neural network. In the first experiment, we included all the $\hat{\beta}_{ext}$ in the training and testing data, and in the second experiment, we removed all instances of $\hat{\beta}_{ext}$ with 0 weight before fitting the model.

Table 5.1 shows the results of two sets of PRS generated per trait. The original NN was the PRS generated using the same configurations as the single model, and the second set of PRS was generated with the same network using the same seed, only that the instances of $\hat{\beta}_{ext} = 0$ was removed from the data prior to model fitting and prediction, and later added back into the PRS for LD adjustment. We observed that the R^2 value of the PRS trained without 0's decreased in R^2 compared to the model trained with the 0 inputs. The PRS for height decreased from 0.2356 to 0.2323, a slight relative decrease of approximately 1.4%. PRS for BMI decreased approximately 3.7% from 0.071 to 0.0684, and showed a more slightly higher decrease in in comparison to height. It appeared that having the $\hat{\beta}_{ext} = 0$ in the data during



(a) Height



(b) BMI

Figure 5.7: Histogram of 0's in the LD adjustment window with 100 SNPs up and downstream for each SNP.

model fitting and prediction stage led to a slight positive improvement in the predictive power of the resulting PRS. We also noted that the change in results is not significant, and the results could differ in different runs due to the random variability in the model fitting. Additionally, due to the use of the SGD optimizer, the change in data could mean a different sample was used in each batch in the model training process, which would also affect the weights of the final NN. Overall, removing the 0's from the training data did not appear to have a significant benefit to the predictive power of the PRS.

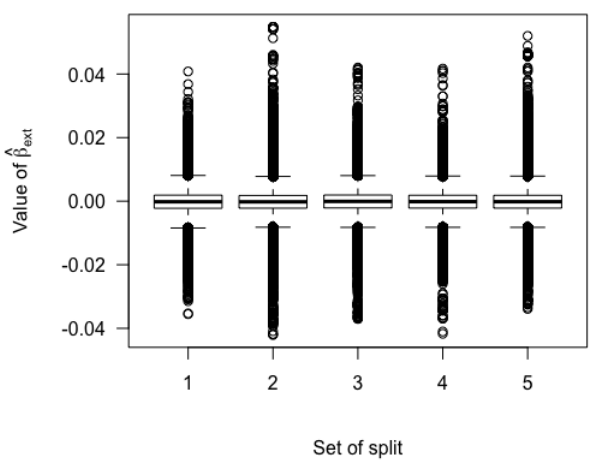
To test the effect of $\hat{\beta}_{ext}$ of 0 on the LD adjusted risk score, we can take a previous set of $\hat{w}$, and remove the 0's prior to performing the LD adjustment. Due to the fact that only <1% of the SNPs have a weight of 0, and that SNPs with weight of 0 should not have high correlation to significant SNPs, we do not expect the effect of the LD adjusted PRS without the SNPs of 0 to have a significant effect on the prediction R^2. Additionally, from the Supplementary Figure S1 of Paré et al. (2017), we observed that varying the window of the LD adjustment did not change the performance of the PRS very much. By removing the SNPs with weight 0, we would expect to replace <1% of the SNPs in the LD adjustment on average, and thus we do not expect the change in R^2 so be significant.

In Figure 5.7, we computed the frequency of the $\hat{w}$ of 0 appearing in the LD adjustment window. For height, we observed 1,937 instances out of approximately 1.98M adjusted weights where the LD adjustment window would have ≥ 5 $\hat{w}$ of 0, corresponding to $< 0.1\%$ of affected SNPs. We only observed 229 instances where the LD adjustment contained ≥ 10 $\hat{w}$ of 0 (representing up to a 5% difference in the LD adjustment calculation), and 10 instances where the LD adjustment ≥ 20 $\hat{w}$ of 0 (representing up to a 10% difference in the LD adjustment calculation). Similarly for BMI, we observed 2,197, 210, and 0 instances of $\geq 5, 10, 20$ $\hat{w}$ of 0 in the LD adjustment process, respectively. Therefore, we can expect to see approximately 0.1% of the weights having a possible change of over 2.5% if we removed any instance of 0 from the LD adjustment process. However, as the SNPs with $\hat{\beta}_{ext}$ of 0 were most likely deemed statistically insignificant to the measured trait, in reality we could expect these SNPs to have very low LD with associated SNPs, thus the overall contribution of the $\hat{w}$ of 0 to the LD adjustment calculation should be very low.

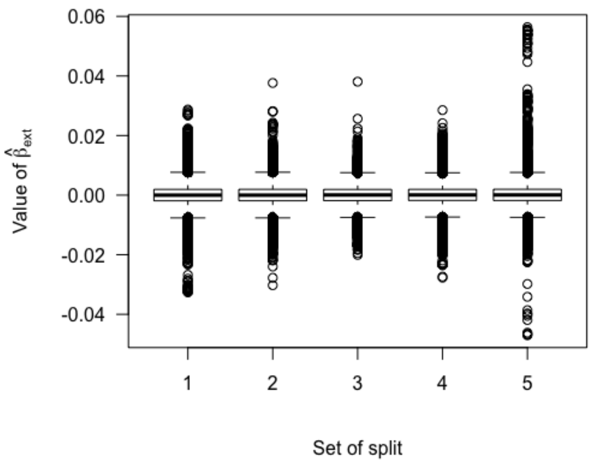
5.3 Difference in Variability in Height and BMI

As can be seen in Figures 4.6 and 4.7, there was a much higher level of variability in the predicted $\hat{w}$ for BMI compared to height from the neural network. Furthermore, as we observed in Figure 4.8, this was a consistent issue among multiple runs of the experiment even with the ensembled models. This represented an issue for the neural network as under the same model architecture and hyperparameters, the resulting PRS for BMI would have had more variable weights than the PRS for height, and by extension, more variable predictive power.

When we decomposed the plot of the mean against the standard deviation of weights across 20 runs with the same model, we observed that the 5th set of predicted $\hat{w}$ (seen in Figure 5.2e was the only set of $\hat{w}$ with a much higher standard deviation across multiple runs than the rest of the splits, reaching approximately 0.007, whereas the rest of the plots in Figure 5.2 displayed a lower much more consistent range of standard deviations when compared to Figure 5.1, with



(a) Height

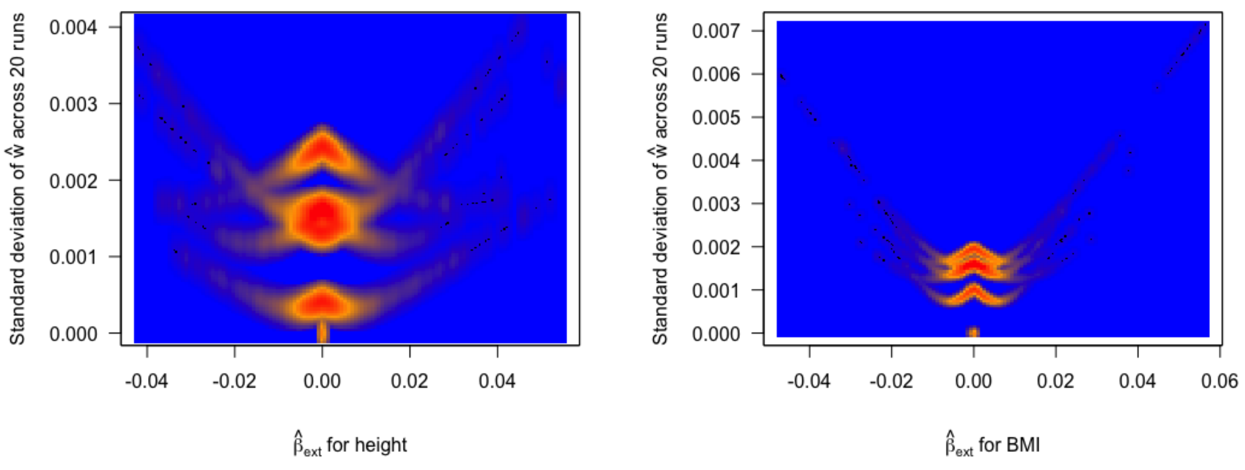


(b) BMI

Figure 5.8: Box plot of $\hat{\beta}_{ext}$ across the 5 splits of the data.

the highest standard deviation still below 0.005.

Figure 5.8 shows a box plot of the $\hat{\beta}_{ext}$ across each of the 5 splits. We observed that in Figure 5.8b, the 5th split displayed a much wider range in $\hat{\beta}_{ext}$ when compared to the rest of the four splits. This suggested that there are certain SNPs in the 5th split that were determined by the external GWAS to have a higher association with BMI than in the rest of the splits. The stronger association with the phenotype in the external GWAS appeared to be correlated with the variability of the predicted $\hat{w}$ in our NN calibrated PRS. As shown in the plot shown in Figure 5.3, the predicted weights for set 5 had a greater variability than the rest of the predicted sets across 20 runs of the model fitting, reaching a high of approximately 0.007 in standard deviation. The source of variation of the $\hat{w}$ in the 5th split for BMI appeared to be connected to the more extreme weights that occur in the 5th split of the $\hat{\beta}_{ext}$. From the GIANT consortium, the most associated SNPs to BMI were found in SNP Set 5, and belonged to chromosome 16. The most positively associated SNP with BMI was determined by the external GWAS to be rs1421085 with a $\hat{\beta}_{ext}$ of 0.0560014, while the most negatively associated SNP, rs3751813, was also found on chromosome 16 with a $\hat{\beta}_{ext}$ of -0.04702976.



(a) Height (b) BMI

Figure 5.9: Plot of the $\hat{\beta}_{ext}$ against the standard deviation of the predicted $\hat{w}$ across 20 experiments for height and BMI

A similar effect of the more extreme $\hat{\beta}_{ext}$ impacting the variability of the predicted weights could be also seen in SNP Set 2 for height. We noticed that in Figure 5.8a, SNP Set 2 appeared to have more extreme values than the rest of the splits. This corresponded to the the behaviour of the weights shown in Figure 5.4a and Figure 5.1 where indeed, SNP Set 2 displayed a higher variability in predicted weights than the rest of the sets. This suggested that there were SNPs in SNP Set 2 of the $\hat{\beta}_{ext}$ splits that were determined by the external GWAS to be more associated with height than others. We observed that all but one SNP with $|\hat{\beta}_{ext}| > 0.05$ where found on chromosome 3 (in Split 2), and one SNP, rs143384, found on chromosome 20 (Split 5). The most positively associated SNP for height was determined to be rs1344674 (found on chromosome 3) with a $\hat{\beta}_{ext}$ of 0.0548436, and the most negatively associated SNP was rs6810223, was also found on chromosome 3, with a $\hat{\beta}_{ext}$ of -0.04210056.

While the most associative SNPs for both height and BMI did have an effect on the variability of the predictions, we also observed that the variability of $\hat{\beta}_{ext}$ between the 5 splits could also contribute to the difference in variability between height and BMI. Figure 5.8b showed a high level of variability between the 5th set of $\hat{\beta}_{ext}$ and the rest of the four splits, as it appeared that the rest of the 4 splits of $\hat{\beta}_{ext}$ were not very associated with BMI. By comparison, in Figure

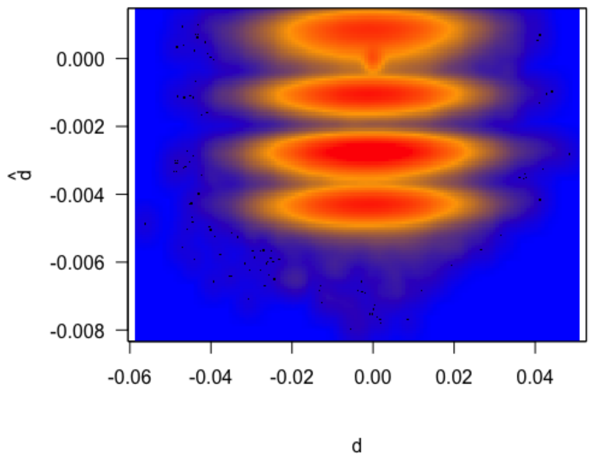
5.8a, there was not as much variation between set 2 of $\hat{\beta}_{ext}$ and the rest of the splits. The higher variability between the most associative split of SNPs and the rest of the sets found in BMI could explain why the predicted $\hat{w}$ for BMI had a had higher variability compared to height.

Figure 5.9 displayed a similar relationship between the $\hat{\beta}_{ext}$ and the standard deviation of the $\hat{w}$ across the 20 experiments. We observed a similar pattern displayed in the plots compared to Figure 4.7. We observed an increase in standard deviation of $\hat{w}$ as the magnitude of $\hat{\beta}_{ext}$ increases, particularly in Set 2 for height and Set 5 for BMI. The figure indicated that the $\hat{\beta}_{ext}$ played a role in the variability of the predictions, particularly for the more extreme values. The correlation between the more extreme $\hat{\beta}_{ext}$ in the data and higher variability in the predictions presented a drawback to using neural networks to calibrate the polygenic risk score. Since the more extreme weights would have a higher effect on the predictive power of the PRS, and that more extreme weights resulted in higher variability after the calibration step, the resulting PRS would have a highly variable predictive power of the trait of interest.

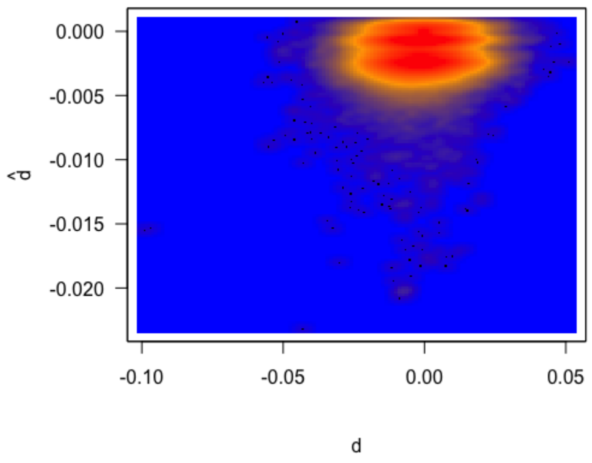
To reduce the variability of the predictions, particularly for BMI, we could employ several potential solutions. A different NN architecture and set of hyperparameters may be used to account for the larger variability in the predicted $\hat{w}$ for BMI. Additionally, we can employ L_1/L_2 regularization and weight decay in the model, or decrease the size of the neural network in order to adjust the bias and decrease variability in our model outputs. Lastly, a larger ensemble may be needed to further reduce the variation in predictions. Due to time constraints, we were not able to fully test the effects of these regularization techniques.

5.4 Effect of Different Splits on the PRS

In earlier sections, we noted that the $\hat{w}$ were clustered into 5 distinct groups based on the splits of the data used in the model fitting. In Figure 5.10, we observed the same distinct clustering of the $\hat{d}$. The effect was particularly noticeable for height (Figure 5.10a). The clusters of $\hat{d}$ indicated that the variability between the sets of predicted $\hat{d}$ was much higher than the variability within each set itself. The clusters of predicted points was also present in the plot of d

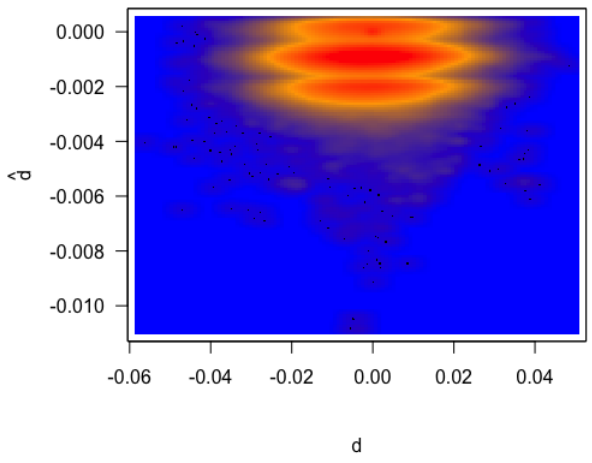


(a) Height

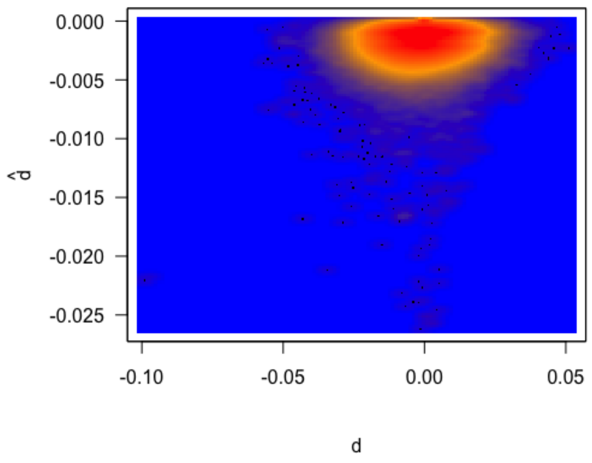


(b) BMI

Figure 5.10: Plot of the dependent variable d against the predicted adjustment factor $\hat{d}$ from the network. The predicted $\hat{d}$ are from a single model.



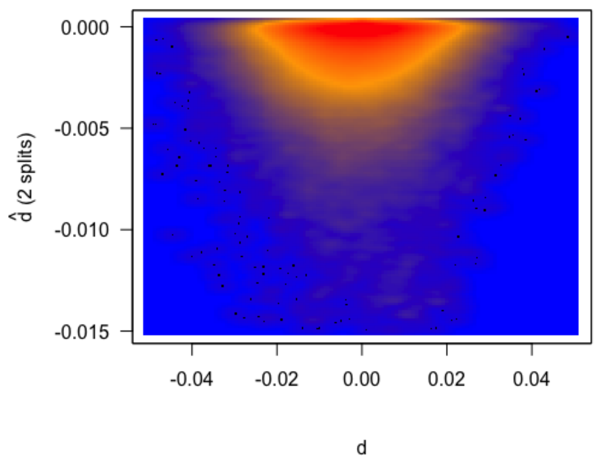
(a) Height



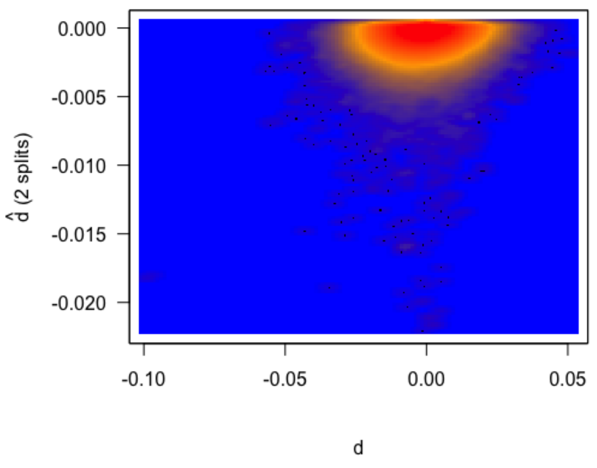
(b) BMI

Figure 5.11: Plot of the dependent variable d against the predicted adjustment factor $\hat{d}$ from the network. The predicted $\hat{d}$ are from the average of 20 models.

against $\hat{d}$ for the averaged output of the 20 models, as seen in Figure 5.11, but to a lesser degree than in the single model. For the ensemble models, we observed overlaps of clusters to a higher degree, particularly in the plots for height in Figure 5.11a, this suggested that the ensemble was shifting the majority of the points closer to 0, further reducing the variability of $\hat{d}$ between



(a) Height

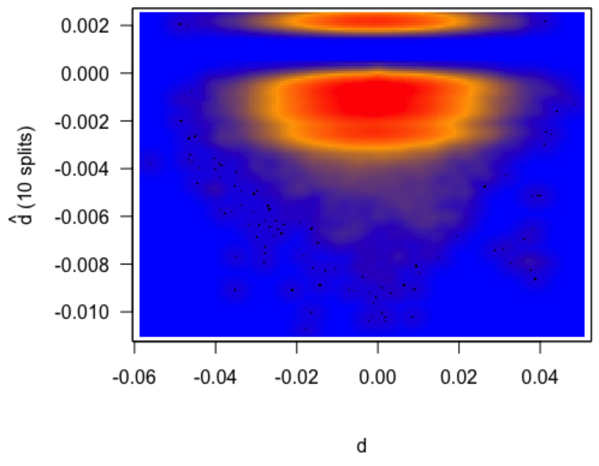


(b) BMI

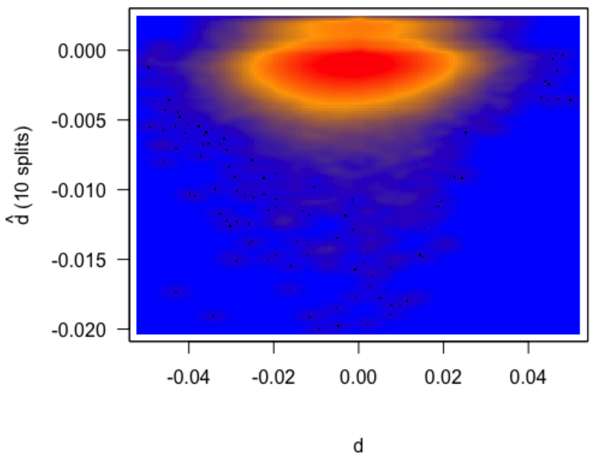
Figure 5.12: Plot of the dependent variable d against the predicted adjustment factor $\hat{d}$ from the network. The predicted $\hat{d}$ are from a single model trained on data divided into 2 contiguous sets.

the distinct sets. In addition to the clusters of $\hat{d}$ based on their predicted sets, we also noticed a cluster of points at (0,0) on the axes. We knew that these points were unaffected by the neural network, as an input $\hat{\beta}_{ext}$ would always result in $\hat{d} = 0$ for the univariate features. We also saw that for both traits, the vast majority of predicted $\hat{d}$ lie below 0, and there appeared to be a wider range of $\hat{d}$ for BMI compared to height, exhibiting consistent behaviour with the increased variability and wider range of weights seen in the PRS for BMI. The distinct clusters of $\hat{d}$ suggested that the predictions were influenced by the way the data is split. This presented an issue with the neural network as Figure 5.10 and 5.11 showed that the greatest factor influencing the predicted $\hat{d}$ is not the $|\hat{\beta}_{ext}|$, but rather the split of the data.

To test the effect of the split in the training data on the final PRS, we experimented with both increasing and decreasing the number of splits in the data. By changing the number of contiguous splits, we hypothesized that the number of distinct clusters would also change to reflect the number of splits. Figure 5.12 depicts the plots of the d against $\hat{d}$ obtained by splitting the data into 2 contiguous splits, then performing the weight calibration with the same neural network architecture as previous experiments. By decreasing the number of splits in the data,



(a) Height

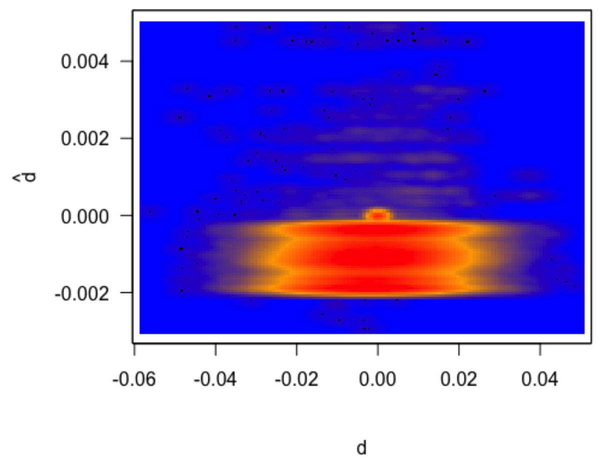


(b) BMI

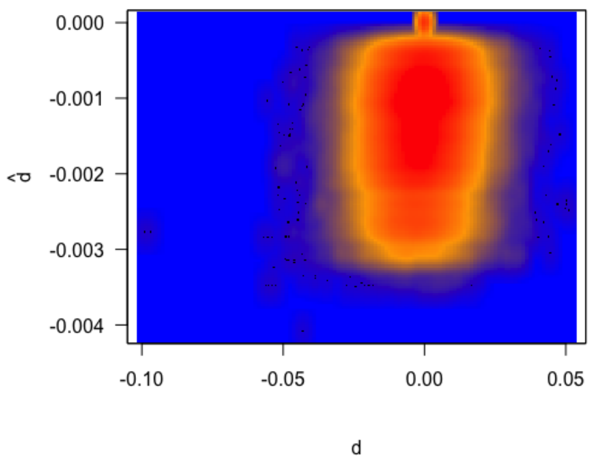
Figure 5.13: Plot of the dependent variable d against the predicted adjustment factor $\hat{d}$ from the network. The predicted $\hat{d}$ are from a single model trained on data divided into 10 contiguous sets.

we noticed that there did not appear to be any distinct clusters in the $\hat{d}$, but there also appeared that for a given value of d, there was a wider range of predicted $\hat{d}$ values. We also saw a negative skew in the $\hat{d}$, indicating a negative adjustment to the $|\hat{\beta}_{ext}|$ and a calibration towards our null hypothesis of no association between the SNPs and the phenotype. Figure 5.13 plotted the d against $\hat{d}$ obtained by splitting the data into 10 contiguous splits. Looking closely at the plots, we still observed several distinct clusters of data, which was particularly noticeable in Figure 5.13a, where one cluster of $\hat{d}$ appeared distinctly above 0, which indicated that the calibrated $\hat{w}$ was further away from 0 compared to $\hat{\beta}_{ext}$ under the formula $(|\hat{\beta}_{ext}| + \hat{d}) \times \text{sign}(\hat{\beta}_{ext})$. We still observed some distinct clusters in Figure 5.13a, but overall there appears to be lower variability between the sets of predicted $\hat{d}$ in both the 2 and 10 splits compared to the initial 5 contiguous splits, with the 2 splits having the lowest variability between sets of $\hat{d}$. The results of the experiment seemed to suggest that splitting the data into 5 contiguous splits seemed to result in higher variability between clusters of predicted $\hat{d}$.

We tested whether the clusters of $\hat{d}$ were the result of using neural networks to perform the calibration, or if it is present in other machine learning algorithms. To test our suspicions,



(a) Height



(b) BMI

Figure 5.14: Plot of the dependent variable d against the predicted adjustment factor $\hat{d}$ from gradient boosted regression trees using 5 splits.

PRS	**Height**	**BMI**
2 Splits	0.1709	0.0597
2 Splits (LD adjustment)	0.2393	**0.0662**
10 Splits	0.1688	0.0562
10 Splits (LD adjustment)	**0.2431**	0.0642

Table 5.2: R^2 Values of different PRS generated by the neural network with with 2 and 10 splits of the data. The best results are highlighted in bold.

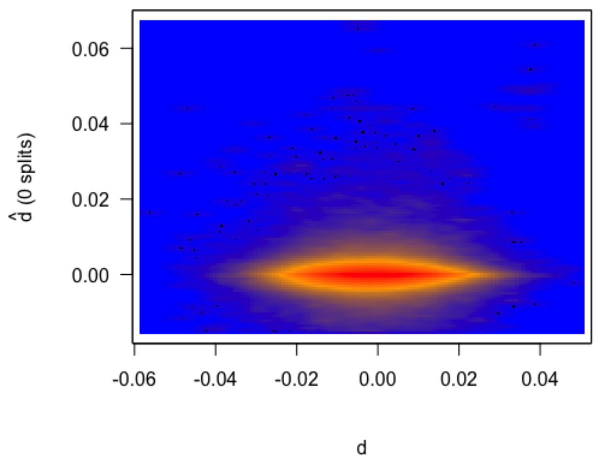
we also plotted the d against $\hat{d}$ obtained using GBRTs. As seen in Figure 5.14, in the plot for height (Figure 5.14a) it also showed multiple clusters of data, although not to the same level of separation as the $\hat{d}$ produced by the neural network. The $\hat{d}$ for BMI produced by the GBRT shows a high degree of overlap between the $\hat{d}$ in each predicted set, depicting very low variability between the sets of $\hat{d}$. We observed that the range of the $\hat{d}$ produced by the GBRT was smaller compared to the $\hat{d}$ produced by the NN. The wider range of $\hat{d}$ predicted by the NN seems to negatively impact the predictive power of the PRS for BMI, but does not seem to affect the predictive power of the PRS for height in a significant manner.

Table 5.2 summarizes the R^2 of the PRS obtained by the neural network trained on 2 and 10 splits. We observed in increase in R^2 for height for both risk scores, and the risk scores were able to outperform the results obtained in Paré et al. (2017) (0.239) and our ensemble model (0.239). However, we see a decrease in prediction R^2 compared to the NN ensemble (0.076). We noted that the change results can be the result of the variability in the neural networks, as discussed in earlier sections, however we did not observe any significant changes from using 2 or 10 splits over the results of the neural network with 5 splits that cannot be explained by the variability in the network.

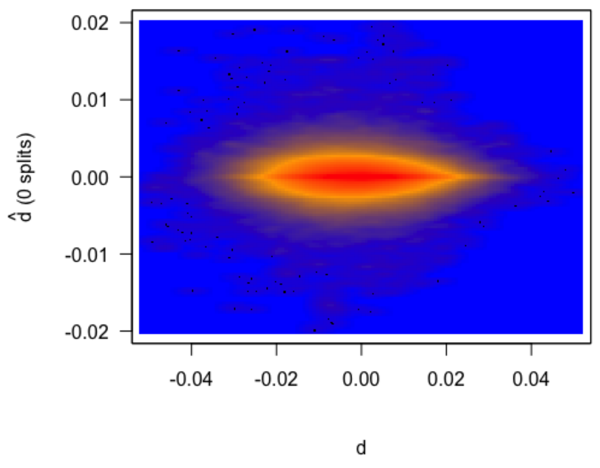
From the plots of d against $\hat{d}$ for both the NN and GBRT, we can draw four conclusions. First, when we used multiple splits to train the models, there appeared to be some variability between each set of predicted $\hat{d}$ present in both neural networks and gradient boosted regression trees. Secondly, the amount of variability between groups appears to be dependent on how heritable a trait is. In other words, for a less heritable trait such as BMI, there is lower separation between the clusters of $\hat{d}$ compared to a highly heritable trait like height. Thirdly, neural networks appear to produce a higher level of variability between the sets of $\hat{d}$ compared to GBRT, at least in the case of 5 splits, and the was less overall variability present in the $\hat{d}$ for GBRT for both traits, which helped improve the predictive power of the PRS for BMI. Despite the lower variability in $\hat{d}$ for GBRT, we still observed variability present. Lastly, the degree of separation between each set of $\hat{d}$ appeared dependent on the number of splits. We observed that for lower splits, there is less variability between sets of $\hat{d}$ compared to higher number of splits such as 5 or 10.

5.5 Using no Splits of the Data to train the NN

We saw in Figure 5.10 and 5.13 that having a higher number of splits resulted in distinct clusters in the predicted $\hat{d}$ based on the splits, and that decreasing the number of splits seemed to decrease the variability between groups (Figure 5.12). We thus experimented with training the neural network with the entirety of the data. To do this, we used the entire set of data to train the neural network, reserving a portion of the data aside to monitor the validation loss. After the neural network is trained, we use all the $|\hat{\beta}_{ext}|$ to predict the calibrated $\hat{w}$. Figure 5.15 shows the plot of d against $\hat{d}$ when the model is trained on all the available data. We



(a) Height



(b) BMI

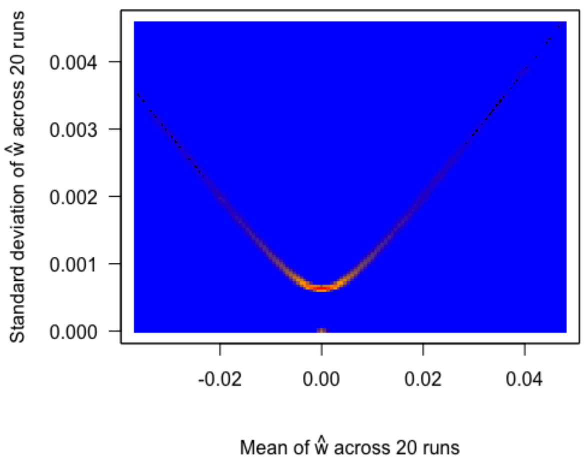
Figure 5.15: Plot of the dependent variable d against the predicted adjustment factor $\hat{d}$ from the network. The predicted $\hat{d}$ are from a single model trained on no splits to predict itself.

PRS	**Height**	**BMI**
Model 1	0.245	0.082
Model 2	0.241	0.0796
Model 3	0.246	0.0803
Model 4	0.238	0.0796
Model 5	0.238	0.0773
Ensembled Model	0.242	0.0804

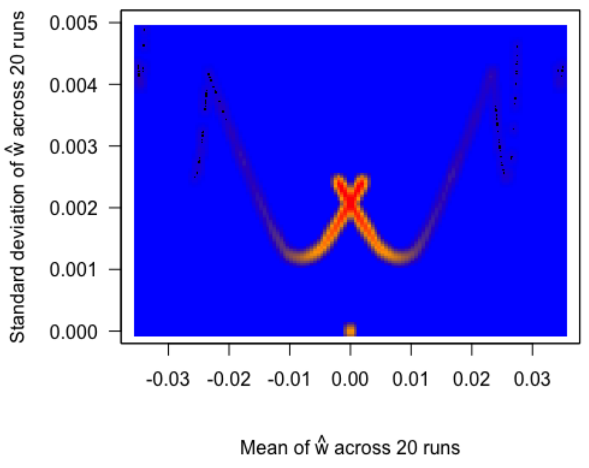
Table 5.3: R^2 Values of different PRS generated by the neural network with with 0 splits after performing LD adjustment.

observe no distinct separation between any clusters of points. It appears that the majority of the predicted $\hat{d}$ are centered around 0.

Table 5.3 summarizes the results of 5 sets of PRS calibrated by 5 neural networks trained on no splits, as well as an average of the calibrated weights of the 5 networks. We observed a substantial increase in the prediction R^2 for BMI, as well as a slight improvement in prediction R^2 for height. The best model obtained an R^2 of 0.246 (Model 3) for height, representing a 2.96% relative increase over the previous best PRS of 0.239. For BMI, we observed that our best model achieved an R^2 of 0.082, matching the predictive power of the GBRT. Our ensemble model achieved an R^2 of 0.242 in height, indicating a 1.26% relative increase over the GBRT, as



(a) Height



(b) BMI

Figure 5.16: Plot of mean against sd of $\hat{w}$ across 20 runs with no splits.

well as an R^2 of 0.0804 for BMI, which was a 5.79% relative increase from our previous best NN calibrated PRS, but did not yield an improvement in prediction R^2 for BMI when compared to GBRT.

Figure 5.16 shows the plot of the mean and standard deviation of $\hat{w}$ across 20 runs of the experiment using no splits. We observed a decrease in variability of the predicted $\hat{w}$ in BMI, decreasing from approximately 0.007 to 0.005, but we did not observe a similar decrease in variability of the predicted weights for height. Most importantly, the predicted weights are no longer grouped into distinct clusters with different relationships between the mean and standard deviation, but rather all the $\hat{w}$, with the exception of the $\hat{w} = 0$, are clearly grouped into one set.

From our experiments using no splits, we observed several distinct advantages of using no splits. First, there was an increase in predictive power of the calibrated PRS using no splits to train the NN. The main reason for dividing the data into k-fold splits is the concern of the model overfitting to the calibration set, thus decreasing the out of sample prediction R^2, however the results showed that this was not the case. By removing the splits, we saw an increase in R^2 for both traits, even when the model architecture and hyperparameters are kept the same. Additionally, from Figure 5.15, it did not appear that the values of $\hat{d}$ were overly influenced by the values of d, and thus did not show any signs of the NN overfitting to the calibration data set. Another distinct advantage of using no splits that we observed was the reduced variability of $\hat{w}$. From Figure 5.16, the variability of the predicted $\hat{d}$ was no greater than in the single models, and in the case of BMI, we saw a decrease of standard deviation from approximately 0.007 to 0.005 when using no splits, which could then be further reduced by model ensembling. The reduced variability is useful for obtaining a stable calibration of the PRS. Lastly, by using no splits, there was a significant increase in the speed of model fitting. When using 5 splits, it took on average 23 seconds per epoch of training. Using no splits, we observed an average of 44 seconds per epoch of training, but with the advantage that we only need to train the model once, instead of 5 times. The speed up in training could greatly benefit prototyping of models, and allow for greater number of experiments within the same time frame. Furthermore, we propose experimentation with GBRT trained with no splits to calibrate the weights of the PRS, as we suspect that it could improve the predictive power of the GBRT as well.

5.6 Interpretability of Neural Networks

One challenge we faced in analyzing the outputs of the neural networks was that due to the cascading structure of stacking linear and non-linear functions, neural networks are often not very interpretable. The neural network architecture thus led to difficulties understanding how the model obtained the results it did. One of the main research areas in neural networks is in understanding how the neural network makes decisions. Recently, there has been work focused on the explainability of neural networks. Some of the issues with with NN interpretability, as well as recent advances, are detailed in Samek et al. (2017). Because it can be hard to to interpret a neural network's outputs, we have limited explanations for some of the behaviours exhibited in the neural network's outputs, such as why the number of splits in the data have such a distinct effect on the output weights, and why there are varying patterns in the relationship between the mean and standard deviation of the predicted weights seen in Figure 5.1 and Figure 5.2.

5.7 Issues With Relying on an External GWAS

While the NN calibrated risk score improved upon the predictive powers of the unadjusted $\hat{\beta}_{ext}$, there were inherent limitations of the PRS due to the way the $\hat{\beta}_{ext}$ were obtained. Since the $\hat{\beta}_{ext}$ were obtained from an external GWAS in the GIANT consortium, we have not addressed several issues pointed out in de Vlaming and Groenen (2015). As stated in earlier chapters, de Vlaming and Groenen (2015) name several issues with the methodology behind conducting a GWAS. Firstly, the GWAS was conducted using one SNP at a time. Secondly, SNPs in LD are not accounted for, leading to possible over-emphasizing of regions of SNPs in high LD. Lastly, the GWAS does not take into account any possible interaction effects between the SNPs. While the LD adjustment from Paré et al. (2017) adjusts the PRS by the pairwise LD between SNPs, we were still only using a machine learning algorithm to calibrate the weights from a GWAS that was performed one SNP at a time, and did not any interaction terms. Unfortunately, these shortcomings cannot be resolved while we rely on an external GWAS for the $\hat{\beta}_{ext}$.

Chapter 6

Conclusion and Future Directions for Research

6.1 Conclusion

In this work, we have conducted numerous experiments on using neural networks to calibrate the weights of a polygenic risk score to a certain target population in order to better improve its predictive capabilities. By using neural networks, we were able to achieve predictive powers that exceeded previous calibration methods, such as P+T and LDPred for both height and BMI, with an R^2 of 0.234 and 0.074, respectively.

In our analysis, we highlighted several issues with using neural networks to calibrate the polygenic risk score. One key issue with using neural networks to perform the weight calibration is the high variance of the calibrated weights across multiple runs, resulting in multiple risk scores with highly varied predictive powers. To address this issue, we proposed using an ensemble of neural networks in order to reduce the variance of the predicted adjustment factors, which resulted in an increase in the prediction R^2, as well as a decreased variance in the performance of the resulting risk scores. The PRS calibrated by the ensemble of neural networks was able to achieve a prediction R^2 of 0.239 and 0.076 for height and BMI, exceeding the performance of PRS a single network.

We also examined the effect of changing the number of splits in the data used to train the models. We observed distinct clusters of predicted $\hat{w}$ made by the neural network associated with the splits of the data. This effect was also present in the predicted $\hat{w}$ calibrated by the gradient boosted regression trees. The presence of distinct groupings of $\hat{w}$ presented a possible downside of using machine learning algorithms to calibrate polygenic risk scores, as it showed that the split of the training data had a much higher effect on the resulting $\hat{w}$ compared to the value of the input data. To remove the effect of the split of the data on the weight calibration, we removed the split in the data, and used the entirety of the data to train a neural network to perform the weight calibration. We observed several advantages of using no training splits in our model fitting, including faster model fitting, reduced variability of predictions, and improved predictive power. An ensemble of neural networks trained on all available data was able to achieve a prediction R^2 of 0.242 for height, outperforming the results obtained in Paré et al. (2017), and 0.0804 on BMI. These results could potentially be extended to other machine learning algorithms such as GBRT to further improve the predictive power of the calibrated PRS.

6.2 Further Model Ensembling

We have already seen that by using an average of predictions made by multiple neural networks, we were able to obtain a more consistent result compared to a single model. Additionally, we saw that a larger ensemble of models yielded greater predictive power than a smaller ensemble of models. Another way to leverage the power of ensemble models would be through the use of *model stacking*. Model stacking was introduced in Wolpert (1992), and it has been a very effective form of model ensembling. Instead of taking a linear combination of the predictions of multiple models, stacking takes the predicted values made by a model (or multiple models), and concatenates the predictions onto the original data as additional features. Then, a new model can be trained on this new data, taking into account the predictions made by models in the previous "layer" of the stack. By incorporating information from previous models into its predictions, the results of a stacked model can often outperform that of a single model, and thus can be used to improve the predictive power of the neural networks. Stacked models have become a very popular method of boosting the predictive power of machine learning models, and play a key role in high-placing solutions in many Kaggle competitions (Töscher et al., 2009; Puurula et al., 2014; Hoch, 2015), and thus we believe that model stacking could play a role in improving the predictive power of our PRS. The stacked model can be further extended to have arbitrarily many layers, with the predictions made by the model(s) in the previous layer feeding into the next layer as concatenated features.

One additional method of model ensembling that we were not able not try, but believe will be very useful is by using an ensemble of multiple different models. Paré et al. (2017) demonstrated that GBRT was able to produce very predictive PRS. It is possible that by incorporating the predicted weights made by the GBRT (either by model averaging or model stacking), we can improve the predictive power of the NN and obtain better results for our PRS. By incorporating models that are less correlated to neural networks, such as GBRTs, we can further reduce the variability of the predictions, leading the more stable predictions as well.

Algorithm 2 1-Layer Model Stacking Algorithm

```
1: Given $D = \{\mathbf{x}_i, y_i\}_{i=1}^n$
2: for $t$ from 1 to $m$ do
3:     Fit a model, $h_j(x)$ to $D$
4:     Take the prediction from model $j$ for $y_i$, $\hat{y}_i^{(j)}$, and concatenate onto $\mathbf{x_i}$ to get $\mathbf{x}_i^{(new)}$
5: Fit a model $h_{m+1}(x)$ to $D_{new} = \left\{\mathbf{x}_i^{(new)}, y_i\right\}_{i=1}^n$
```

6.3 Novel Model Architectures

While there has been a lot of advancement in new neural network architectures in recent years, much of the progress has been made in computer vision and natural language processing. Feed forward neural networks (FFNN), by contrast, have not seen major advancements in model architectures. While there has been limited recent developments in FFNN architecture, we observe several advancements for convolutional neural networks (CNNs) that can be adapted to work for FFNN, and could lead to improvements in predictive powers for our neural networks.

So far we have seen that a shallower Neural Network tends to outperform deeper networks in predictive power. To fit a deeper network to our data, we can take a look at recent advances in computer vision for inspiration. Residual networks (resnets), first proposed in He et al. (2016a), showed that by taking "gradient shortcuts", deeper models can be built while preserving the gradient flow of shallower networks, which allowed for faster model convergence and improved predictive power of deeper networks. This idea was further extended by Huang et al. (2017), where the input for one convolutional layer in the network is directly concatenated onto the output, so that the next layer in the network receives its inputs from all previous output layers in the network. Huang et al. (2017) argued that by using this connection between layers, information in the preceding layers in the NN is directly used by the later layers, thereby allowed the model to gain more information from previous layers when compared to the gradient shortcuts used in resnets. By leveraging the core concept from Huang et al. (2017) and applying it to FFNNs in a similar fashion, we propose using a feed forward neural network where the inputs are concatenated directly to the outputs of that layer. By using this type of network, it should theoretically allow us to build deeper networks while preserving the predictive power of the shallower network.

We can also take into consideration recent advances in CNN model architecture by using a *pre-activation* model as described in He et al. (2016b). In a pre-activation model, instead of layers following a Linear, ReLU, Batchnorm, Dropout structure, we instead use Batchnorm, ReLU, Dropout, Linear. He et al. (2016b) showed that a pre-activation resnet resulted in an increase in model performance when compared to the traditional resnet. Empirically, pre-activation networks have shown to have a better predictive capabilities compared to the traditional architecture, and is the current adopted method for designing CNNs. One shortcoming of the pre-activation architecture is that it was mainly designed for CNNs, and there have been no studies we could find on its usage in FFNNs, However, due to the ease of implementation, we believe that this could be a useful next step in optimizing neural networks for PRS.

6.4 Convolutional and Recurrent Neural Networks

CNNs have seen success in certain applications in the field of genetics, as seen in Alipanahi et al. (2015), Zeng et al. (2016), and Zhou and Troyanskaya (2015). One advantage CNNs have over FFNNs is that the use of convolutional filters is able to take into account any relationships between the $\hat{\beta}_{ext}$ of the input SNP and the $\hat{\beta}_{ext}$ of all other SNPs in the convolutional filter. We hypothesize that due to the correlated nature of SNPs, CNNs may be particularly suited to handle this type of data. In this case, it may be possible for CNNs take into account the relationships between a SNP and its neighbors on the genome, such as linkage disequilibrium or any synergistic/antagonistic effects, and factor that into its output adjustment factors, possibly improving the predictive power of the resulting PRS. We propose that future studies could look into the predictive power of a 1-dimensional CNN for predicting polygenic traits. In order to leverage the possible relationship between a SNP and its surrounding SNPs, each value of $\mathbf{Z_j}$, which is currently a univariate vector of the absolute value of the regression coefficient for SNP j can be concatenated with L regression coefficients of SNPs before and after it. The resulting feature, $\mathbf{Z_j}$, would then be a vector of length $2L + 1$. This feature vector can then be used as the input to a CNN with 1-dimensional convolutional layers with the corresponding adjustment factor for SNP j as the dependent variable.

Using CNNs for computing PRS could also have several challenges over using the FFNN. One possible downside of using 1D CNNs would be that as the convolutional kernels convolves over a sequence, it reuses input features multiple times. Because the convolutional kernel is fixed in length, the regression coefficients in the first and last L of each sequence would not be reused as many times as the other features. Additionally, as SNPs in one chromosome are not affected by SNPs in another chromosome, some way of sequence padding will be needed so that the convolutional kernel will not be using $\hat{\beta}_{ext}$ from two different chromosomes in its predictions. By using a neural network that can take into account possible influences from surrounding SNPs, the dataset needs to be split into training and prediction sets in such a way that SNPs in chromosomes are not split into multiple sets, as opposed to the current method of splitting into 5 contiguous sets.

Similar to CNNs, we have observed recurrent neural networks (RNNs) successfully applied to other fields in genetics because of their ability to learn sequence dependencies in the features of given genetic data, such as Hill et al. (2017). Because 1-dimensional CNNs and RNNs have both shown predictive power for sequential data, we believe that a RNN may also be able to outperform the current neural networks. Similar to 1-dimensional CNNs, we propose that for a given SNP j, the input vector $\mathbf{Z_j}$ should be of length $2L+1$, taking the L summary statistics before and after that SNP, with the target variable being the observed d for SNP j. One key difference of using RNNs instead of CNNs is that in recurrent neural networks, each input in the sequence is used only once (or twice if using a bi-directional RNN), thus we do not reuse any inputs in a given forward pass. However, the first and last L SNPs for a chromosome would still require some form of sequence padding to ensure that the input vectors are of the same length. Additionally, much like their CNN counterpart, a RNN will still require a strategic split in the data, as well as sequence padding between chromosomes as described in the proposed strategy for implementing CNNs.

6.5 Sequence to Sequence Networks

Another novel approach to calculating the polygenic risk score could be through the use of *sequence to sequence* (seq2seq) networks. Seq2seq networks present possible solutions to some of the problems posed by RNNs and CNNs. First introduced in Sutskever et al. (2014), seq2seq networks are the only neural network architecture we could find in literature that can be fitted to sequential data with variable length features and variable length outputs, which means that we can use one sequence to make multiple predictions. Seq2seq networks functions by using two RNNs. The first RNN (called the encoder), takes in data with a variable feature length and outputs a hidden state h_t, which is fed into the second RNN (called decoder) and returns an output that is also variable in length. With recent advances in seq2seq learning, such as the use of the *attention mechanism* first introduced in Vaswani et al. (2017), combined with bi-directional LSTM (Graves and Schmidhuber, 2005) and bi-directional GRUs (Bahdanau et al., 2014), seq2seq networks have been widely used in sequence to sequence tasks such as machine translation. We propose that in order to leverage seq2seq networks, the external regression coefficients are first ordered corresponding to their SNP location and split into sequences of a pre-specified length, L. The regression coefficients within each sequence are then fed into the seq2seq network as a sequence of $\hat{\beta}_{ext}$'s, and the output is a sequence of adjustment factors, $\hat{d}$'s which is of the same length as the input.

Due to the fact that seq2seq networks can take a variable length sequence of $\hat{\beta}_{ext}$ as inputs and compute a corresponding vector of $\hat{d}$'s, we no longer the need to pad the inputs if we are at the beginning and the end of a chromosome. For example, using a CNN/RNN, at the 1^{st} SNP of a chromosome, we would need to include padding for the first L of the sequence, so to ensure that the corresponding $\hat{\beta}_{ext}$ appears at the center of the input vector. However, using a seq2seq network, we just need to input the a vector of $\hat{\beta}_{ext}$'s without padding to get an output of corresponding $\hat{d}$'s. A similar approach could be used at the end of each chromosome. we can stop the sequence at the end of each chromosome to ensure that the SNPs in each chromosome remain independent of each other, thus avoiding the issues with padding between chromosomes present for RNNs and CNNs.

A seq2seq network could outperform our current neural networks in predictive power as they can leverage not only the $\hat{\beta}_{ext}$ for a given SNP, but also all the other $\hat{\beta}_{ext}$'s in that given sequence (or the entire chromosome). With the attention mechanism, the network could potentially learn to place emphasis on which SNPs could contribute more (or less) to a certain phenotype. So far, we were not able to find any instances in literature of a seq2seq network applied directly to SNP data. However, we believe that a network that leverages more information from surrounding SNPs could outperform a network that uses information from just one given SNP individually. Using seq2seq networks would also require the data be split in a way such that no one chromosome will appear in multiple splits.

Appendix A

Appendix

A.1 Common Activation Functions

Sigmoid: $f(x) = \frac{1}{1+e^{-x}}$

ReLU:$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$

TanH: $f(x) = \frac{(e^{x}-e^{-x})}{(e^{x}+e^{-x})}$

A.2 Common Loss Functions

Binary Classification: Binary Cross-entropy

$$J = -(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i))$$

Where y_i is a binary label 0/1 for observation i and $\hat{y}_i$ is the predicted probability of input $\mathbf{x_i}$ belonging to class y_i.

Multi-class Classification: Categorical Cross Entropy with K classes

$$J = -\sum_{c=1}^{K} y_{i,c} \log(\hat{y}_{i,c})$$

Where $y_{i,c}$ is a label with value $\in \{1, 2, \ldots, c\}$ indicating that observation i belongs to class c and $\hat{y}_{i,c}$ is the predicted probability of input $\mathbf{x_i}$ belonging to class c.

Regression: Mean Square Error

$$J = \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N}$$

Regression: Mean Absolute Error

$$J = \frac{\sum_{i=1}^{N}|y_i - \hat{y}_i|}{N}$$

While the Mean Absolute Error function is not differentiable at $y_i = \hat{y}_i$, in most software implementations (such as Tensorflow), the loss function is often evaluated to be 0.

A.3 Optimization Algorithms

SGD with Momentum

Initialize with $\gamma < 1$ (typically set at 0.9 or similar value). (Ruder, 2016)

Algorithm 3 SGD with Momentum

```
1: Initialize $v^{(0)}$, $\alpha$, and $\gamma$
2: for $t$ from 1 to $T$ epochs do
3:     $v^{(t)} = \gamma v^{(t-1)} + \alpha \nabla J(W^{(t-1)})$
4:     Weight Update $W^{(t)} = W^{(t-1)} - v^{(t)}$
```

SGD with Nesterov Momentum

Initialize with $\gamma < 1$ (typically set at 0.9 or similar value). (Ruder, 2016)

RMSprop

Algorithm 4 SGD with Nesterov Momentum

```
1: Initialize $v^{(0)}$, $\alpha$, and $\gamma$
2: for $t$ from 1 to $T$ epochs do
3:     $v^{(t)} = \gamma v^{(t-1)} + \alpha \nabla J(W^{(t-1)} - \gamma v^{(t-1)})$
4:     Weight Update $W^{(t)} = W^{(t-1)} - v^{(t)}$
```

Initialize with $\gamma = 0.9$, $\alpha = 0.001$ (Hinton et al., 2012). Denote $g_t \equiv \nabla J(W^{(t-1)})$, and $g_t^2 \equiv g_t \odot g_t$, the element-wise square of g_t.

Algorithm 5 RMSProp

```
1: for $t$ from 1 to $T$ epochs do
2:     Compute running average, $E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2$
3:     Weight Update $W^{(t)} = W^{(t-1)} - \frac{\alpha}{\sqrt{E[g^2]_t + \epsilon}} g_t$
```

Adam

Initialize with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\alpha = 0.001$, and $\epsilon = 10^{-8}$ (Kingma and Ba, 2014). Denote $g_t \equiv \nabla J(W^{(t-1)})$, and $g_t^2 \equiv g_t \odot g_t$, the element-wise square of g_t.

Algorithm 6 Adam

```
1: Initialize $m^{(0)}$, $v^{(0)}$
2: for $t$ from 1 to $T$ epochs do
3:     $m^{(t)} = \beta_1 m^{(t-1)} + (1-\beta_1) g_t$
4:     $v^{(t)} = \beta_2 v^{(t-1)} + (1-\beta_2) g_t^2$
5:     $\hat{m}^{(t)} = \frac{m^{(t)}}{1-\beta_1^t}$
6:     $\hat{v}^{(t)} = \frac{v^{(t)}}{1-\beta_2^t}$
7:     Weight Update $W^{(t)} = W^{(t-1)} - \frac{\alpha}{\sqrt{\hat{v}^{(t)}} + \epsilon} \hat{m}^{(t)}$
```

A.4 Batch Normalization Algorithm

Given batch of $B = \{\mathbf{x_i}\}_{i=1}^n$ such that $n \leqslant N$ and parameters γ, β (scale and shift factors to be learned through the backward pass), and $\epsilon = 1 \times 10^{-5}$ (to avoid divide by 0 error).

Algorithm 7 Batchnorm Algorithm

```
1: Define $\mu_{\mathbf{B}} \equiv \frac{1}{n}\sum_{i=1}^{n} \mathbf{x_i}$ and $\sigma_B^2 \equiv \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x_i} - \mu_{\mathbf{B}})^2$
2: Calculate $\hat{\mathbf{x}}_{\mathbf{i}} = \frac{\mathbf{x_i} - \mu_{\mathbf{B}}}{\sqrt{\sigma_B^2 + \epsilon}}$
3: return $y_i = \gamma\hat{\mathbf{x}}_{\mathbf{i}} + \beta$
```

A.5 Batch Normalization Backpropagation

Since batchnorm is applied to the neurons before the activation function, we can consider the batchnorm as a separate layer between the feed-forward layer and the activation layer where for the i-th data point, x_i^ℓ is the $m^\ell \times 1$ input and y_i^ℓ is the $m^\ell \times 1$ output of the batchnorm layer ℓ. We can calculate the gradient of the loss with respect to γ and β as

$$\frac{\partial J(W)}{\partial \gamma} = \sum_{i=1}^{m^\ell} \frac{\partial J(W)}{\partial y_i}\frac{\partial y_i}{\partial \gamma}$$

$$\frac{\partial J(W)}{\partial \beta} = \sum_{i=1}^{m^\ell} \frac{\partial J(W)}{\partial y_i}\frac{\partial y_i}{\partial \beta}$$

Recall that since the output of one layer becomes the input of the next layer, $\frac{\partial J(W)}{\partial y_i^{(\ell)}} = \frac{\partial J(W)}{\partial x_i^{(\ell+1)}}$ which is computed from the previous step of the backprop algorithm, thus $\sum_{i=1}^{m^\ell} \frac{\partial J(W)}{\partial y_i}$ can be assumed to be given. Observe that since $y_i = \gamma\hat{x}_i + \beta$, $\frac{\partial y_i}{\partial \gamma} = \hat{x}_i$ and $\frac{\partial y_i}{\partial \beta} = 1$

$$\frac{\partial J(W)}{\partial \gamma} = \sum_{i=1}^{m^\ell} \frac{\partial J(W)}{\partial y_i}\hat{x}_i$$

$$\frac{\partial J(W)}{\partial \beta} = \sum_{i=1}^{m^\ell} \frac{\partial J(W)}{\partial y_i} \tag{A.1}$$

Which provides the gradients for the weight update step in the backward pass.

In order to compute $\dfrac{\partial J(W)}{\partial x_i}$, we first recognize that μ_B, σ_B^2, and $\hat{x}_i$ all contain x_i, thus we have

$$\frac{\partial J(W)}{\partial x_i} = \frac{\partial J(W)}{\partial \hat{x}_i}\frac{\partial \hat{x}_i}{\partial x_i} + \frac{\partial J(W)}{\partial \mu_B}\frac{\partial \mu_B}{\partial x_i} + \frac{\partial J(W)}{\partial \sigma_B^2}\frac{\partial \sigma_B^2}{\partial x_i}$$

By applying chain rule to each term of the summation, we can compute each term seperately

$$\begin{aligned}\frac{\partial J(W)}{\partial \hat{x}_i}\frac{\partial \hat{x}_i}{\partial x_i} &= \frac{\partial J(W)}{\partial y_i}\frac{\partial y_i}{\partial \hat{x}_i}\frac{\partial}{\partial x_i}\left(\frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}\right)\\ &= \frac{\partial J(W)}{\partial y_i}\gamma\frac{1}{\sqrt{\sigma_B^2 + \epsilon}} \text{ since } y_i = \gamma\hat{x}_i + \beta \qquad \text{(A.2)}\end{aligned}$$

Observe that since σ_B^2 contains μ_B

$$\frac{\partial J(W)}{\partial \mu_B} = \sum_{i=1}^{m^\ell}\frac{\partial J(W)}{\partial \hat{x}_i}\frac{\partial \hat{x}_i}{\partial \mu_B} + \frac{\partial J(W)}{\partial \sigma_B^2}\frac{\partial \sigma_B^2}{\partial \mu_B}$$

Where

$$\begin{aligned}\frac{\partial \sigma_B^2}{\partial \mu_B} &= \frac{\partial}{\partial \mu_B}\left(\frac{1}{m^\ell}\sum_{i=1}^{m^\ell}(x_i - \mu_B)^2\right)\\ &= \frac{1}{m^\ell}\sum_{i=1}^{m^\ell}2(x_i - \mu_B)(-1)\\ &= \frac{-2}{m^\ell}\sum_{i=1}^{m^\ell}(x_i - \mu_B)\\ &= \frac{-2}{m^\ell}\sum_{i=1}^{m^\ell}x_i - m^\ell\mu_B\\ &= 0\end{aligned}$$

So the equation simplifies to

$$\begin{aligned}
\frac{\partial J(W)}{\partial \mu_B} &= \sum_{i=1}^{m^\ell} \frac{\partial J(W)}{\partial \hat{x}_i} \frac{\partial \hat{x}_i}{\partial \mu_B} \\
&= \sum_{i=1}^{m^\ell} \frac{\partial J(W)}{\partial y_i} \gamma \frac{\partial \hat{x}_i}{\partial \mu_B} \text{ from equation (A.2)} \\
&= \sum_{i=1}^{m^\ell} \frac{\partial J(W)}{\partial y_i} \gamma \frac{\partial}{\partial \mu_B} \left(\frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \right) \\
&= \sum_{i=1}^{m^\ell} \frac{\partial J(W)}{\partial y_i} \gamma \frac{-1}{\sqrt{\sigma_B^2 + \epsilon}} \\
&= -\frac{\gamma}{\sqrt{\sigma_B^2 + \epsilon}} \sum_{i=1}^{m^\ell} \frac{\partial J(W)}{\partial y_i} \\
&= -\frac{\gamma}{\sqrt{\sigma_B^2 + \epsilon}} \frac{\partial J(W)}{\partial \beta} \text{ By equation (A.1)}
\end{aligned}$$

Additionally, it can be seen that

$$\begin{aligned}
\frac{\partial \mu_B}{\partial x_i} &= \frac{\partial}{\partial x_i} \frac{1}{m^\ell} \sum_{i=1}^{m^\ell} x_i \\
&= \frac{1}{m^\ell}
\end{aligned}$$

So we have

$$\frac{\partial J(W)}{\partial \mu_B} \frac{\partial \mu_B}{\partial x_i} = -\frac{\gamma}{m^\ell \sqrt{\sigma_B^2 + \epsilon}} \frac{\partial J(W)}{\partial \beta} \tag{A.3}$$

For the last term, we can calculate $\frac{\partial J(W)}{\partial \sigma_B^2}$

$$\frac{\partial J(W)}{\partial \sigma_B^2} = \sum_{i=1}^{m^\ell} \frac{\partial J(W)}{\partial \hat{x}_i} \frac{\partial \hat{x}_i}{\partial \sigma_B^2}$$

The gradient $\frac{\partial \hat{x}_i}{\partial \sigma_B^2}$ can be computed as

$$\begin{aligned}\frac{\partial \hat{x}_i}{\partial \sigma_B^2} &= \frac{\partial}{\partial \sigma_B^2}\left(\frac{x_i-\mu_B}{\sqrt{\sigma_B^2+\epsilon}}\right)\\ &= (x_i-\mu_B)(-\frac{1}{2})(\sigma_B^2+\epsilon)^{-3/2}\end{aligned}$$

Thus

$$\begin{aligned}\frac{\partial J(W)}{\partial \sigma_B^2} &= \sum_{i=1}^{m^\ell}\left(\frac{\partial J(W)}{\partial y_i}\gamma(x_i-\mu_B) - \frac{(\sigma_B^2+\epsilon)^{-3/2}}{2}\right)\\ &= -\frac{\gamma(\sigma_B^2+\epsilon)^{-3/2}}{2}\sum_{i=1}^{m^\ell}\left(\frac{\partial J(W)}{\partial y_i}(x_i-\mu_B)\right)\end{aligned}$$

We also recognize that

$$\begin{aligned}\frac{\partial \sigma_B^2}{\partial x_i} &= \frac{\partial}{\partial x_i}\left(\frac{1}{m^\ell}\sum_{i=1}^{m^\ell}(x_i-\mu_B)^2\right)\\ &= \frac{2(x_i-\mu_B)}{m^\ell}\end{aligned}$$

So we get

$$\begin{aligned}\frac{\partial J(W)}{\partial \sigma_B^2}\frac{\partial \sigma_B^2}{\partial x_i} &= -\frac{\gamma(\sigma_B^2+\epsilon)^{-3/2}}{m^\ell}\sum_{i=1}^{m^\ell}\left(\frac{\partial J(W)}{\partial y_i}(x_i-\mu_B)\right)(x_i-\mu_B)\\ &= -\frac{\gamma(\sigma_B^2+\epsilon)^{-1/2}}{m^\ell}\sum_{i=1}^{m^\ell}\left(\frac{\partial J(W)}{\partial y_i}(x_i-\mu_B)\right)\frac{x_i-\mu_B}{\sigma_B^2+\epsilon}\end{aligned}$$

Observe that $x_i \ - \mu_B = \hat{x}_i\sqrt{\sigma_B^2 + \epsilon}$, so then

$$\begin{aligned}
\frac{\partial J(W)}{\partial \sigma_B^2}\frac{\partial \sigma_B^2}{\partial x_i} &= -\frac{\gamma(\sigma_B^2+\epsilon)^{-1/2}}{m^\ell}\sum_{i=1}^{m^\ell}\left(\frac{\partial J(W)}{\partial y_i}\hat{x}_i\sqrt{\sigma_B^2+\epsilon}\right)\frac{x_i-\mu_B}{\sigma_B^2+\epsilon} \\
&= -\frac{\gamma(\sigma_B^2+\epsilon)^{-1/2}}{m^\ell}\sum_{i=1}^{m^\ell}\left(\frac{\partial J(W)}{\partial y_i}\hat{x}_i\right)\frac{x_i-\mu_B}{\sqrt{\sigma_B^2+\epsilon}} \\
&= -\frac{\gamma(\sigma_B^2+\epsilon)^{-1/2}}{m^\ell}\frac{\partial J(W)}{\partial \gamma}\hat{x}_i \\
&= -\frac{\gamma}{m^\ell\sqrt{\sigma_B^2+\epsilon}}\frac{\partial J(W)}{\partial \gamma}\hat{x}_i \qquad \text{(A.4)}
\end{aligned}$$

Finally, combining equations (A.2), (A.3), and (A.4), we have

$$\frac{\partial J(W)}{\partial x_i} = -\frac{\gamma}{m^\ell\sqrt{\sigma_B^2+\epsilon}}\left[m\frac{\partial J(W)}{\partial y_i} + \frac{\partial J(W)}{\partial \beta} + \hat{x}_i\frac{\partial J(W)}{\partial \gamma}\right] \qquad .$$

A.6 How Model Averaging Reduces Variance

Algorithm 8 Model Averaging Algorithm

```
1: Given $D = \{\mathbf{x_i}, y_i\}_{i=1}^N$
2: for $t$ from 1 to $m$ do
3:     Fit a model, $h_j(x)$ to $D$
4: Final Model $h(x) = \frac{1}{m}\sum_{j=1}^m h_j(x)$
```

If we suppose that each model $h_j(x)$ makes error ε_j, and that $\varepsilon_j \sim N(0,1)$. We can compute the variance and covariance as

$$\begin{aligned}
\varepsilon_i\varepsilon_i &= \text{Var}(h_i(x)), \text{ and} \\
\varepsilon_i\varepsilon_j &= \text{Cov}(h_i(x), h_j(x))
\end{aligned}$$

If we assume that each of the m models are perfectly uncorrelated, such that $\text{Cov}(h_i(x), h_j(x)) = 0$, then:

$$\begin{aligned}\mathrm{Var}(h(x)) = \frac{1}{m^2}\sum_i\sum_j \varepsilon_i\varepsilon_j &= \frac{1}{m^2}\sum_{\substack{i=1\\i=j}}^{m}\sum_{j=1}^{m}\varepsilon_i\varepsilon_j + \frac{1}{m^2}\sum_{\substack{i=1\\i\neq j}}^{m}\sum_{j=1}^{m}\varepsilon_i\varepsilon_j\\ &= \frac{1}{m^2}[m\mathrm{Var}(h_i(x)) + m(m-1)\mathrm{Cov}(h_i(x),h_j(x))]\\ &= \frac{1}{m}\mathrm{Var}(h_i(x)) + \frac{m-1}{m}\mathrm{Cov}(h_i(x),h_j(x))\end{aligned}$$

$$\begin{aligned}\mathrm{Var}(h(x)) = \frac{1}{m^2}\sum_i\sum_j \varepsilon_i\varepsilon_j &= \frac{1}{m^2}\sum_{\substack{i=1\\i=j}}^{m}\sum_{j=1}^{m}\varepsilon_i\varepsilon_j + \frac{1}{m^2}\sum_{\substack{i=1\\i\neq j}}^{m}\sum_{j=1}^{m}\varepsilon_i\varepsilon_j\\ &= \frac{1}{m^2}[m\mathrm{Var}(h_i(x)) + m(m-1)\mathrm{Cov}(h_i(x),h_j(x))]\\ &= \frac{1}{m}\mathrm{Var}(h_i(x)) + \frac{m-1}{m}\mathrm{Cov}(h_i(x),h_j(x))\end{aligned}$$

Since the models are uncorrelated, the covariance between models is 0, thus we get

$$\begin{aligned}\frac{1}{m^2}\sum_i\sum_j \varepsilon_i\varepsilon_j &= \frac{1}{m}\mathrm{Var}(h_i(x)) + \frac{m-1}{m}0\\ &= \frac{1}{m}\mathrm{Var}(h_i(x))\end{aligned}$$

In practice, it is not reasonable to assume that the models are perfectly uncorrelated. However, since the models are producing different predictions for $\hat{d}$, we can see that the neural networks are not perfectly correlated. Thus we can assume that $\mathrm{Cov}(h_i(x), h_j(x)) \leq \mathrm{Var}(h_i(x))$ and that ensembling more models should decrease the variance in predictions.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. (2016). Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283.

Alipanahi, B., Delong, A., Weirauch, M. T., and Frey, B. J. (2015). Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. *Nature Biotechnology*, 33(8):831.

Allen, H. L., Estrada, K., Lettre, G., Berndt, S. I., Weedon, M. N., Rivadeneira, F., Willer, C. J., Jackson, A. U., Vedantam, S., Raychaudhuri, S., et al. (2010). Hundreds of variants clustered in genomic loci and biological pathways affect human height. *Nature*, 467(7317):832–838.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bhandare, A., Bhide, M., Gokhale, P., and Chandavarkar, R. (2016). Applications of convolutional neural networks. *International Journal of Computer Science and Information Technologies*, 7(5):2206–2215.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In HD, P.-V., editor, *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

Bulik-Sullivan, B. K., Loh, P.-R., Finucane, H. K., Ripke, S., Yang, J., Patterson, N., Daly, M. J., Price, A. L., Neale, B. M., of the Psychiatric Genomics Consortium, S. W. G., et al.

(2015). Ld score regression distinguishes confounding from polygenicity in genome-wide association studies. *Nature genetics*, 47(3):291–295.

Cao, C., Liu, F., Tan, H., Song, D., Shu, W., Li, W., Zhou, Y., Bo, X., and Xie, Z. (2018). Deep learning and its applications in biomedicine. *Genomics, Proteomics & Bioinformatics*, 16(1):17–32.

Chen, Y., Li, Y., Narayan, R., Subramanian, A., and Xie, X. (2016). Gene expression inference with deep learning. *Bioinformatics*, 32(12):1832–1839.

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

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Ciresan, D., Giusti, A., Gambardella, L. M., and Schmidhuber, J. (2012). Deep neural networks segment neuronal membranes in electron microscopy images. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, pages 2843–2851. Curran Associates Inc.

De Brébisson, A., Simon, É., Auvolat, A., Vincent, P., and Bengio, Y. (2015). Artificial neural networks applied to taxi destination prediction. *arXiv preprint arXiv:1508.00021*.

de Vlaming, R. and Groenen, P. J. (2015). The current and future use of ridge regression for prediction in quantitative genetics. *BioMed Research International*, 2015:e143712.

Dudbridge, F. (2013). Power and predictive accuracy of polygenic risk scores. *PLoS Genetics*, 9(3):e1003348.

Eldan, R. and Shamir, O. (2016). The power of depth for feedforward neural networks. In Feldman, V., Rakhlin, A., and Shamir, O., editors, *Conference on Learning Theory*, pages 907–940. PMLR.

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., and Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Teh, Y. W. and Titterington, M., editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256. PMLR.

Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Güneş, F., Wolfinger, R., and Tan, P.-Y. (2017). Stacked ensemble models for improved prediction accuracy. `https://support.sas.com/resources/papers/proceedings17/SAS0437-2017.pdf`. Accessed: 2018-05-30.

Guo, C. and Berkhahn, F. (2016). Entity embeddings of categorical variables. *arXiv preprint arXiv:1604.06737*.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on Imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034. IEEE.

He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778. IEEE.

He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer.

Hill, S. T., Kuintzle, R. C., Teegarden, A., Merrill, E., Danaee, P., and Hendrix, D. A. (2017). A deep recurrent neural network discovers complex biological rules to decipher RNA protein-coding potential. *bioRxiv: 200758*.

Hinton, G., Srivastava, N., and Swersky, K. (2012). Neural networks for machine learning Lecture 6a overview of mini-batch gradient descent. `https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`. Online; Accessed: 2018-05-30.

Hoch, T. (2015). An ensemble learning approach for the Kaggle taxi travel time prediction challenge. In Martínez-Usó, A., Mendes-Moreira, J., Moreira-Matias, L., Kull, M., and Lachiche, N., editors, *Proceedings of the 2015th International Conference on ECML PKDD Discovery Challenge*. CEUR-WS.org.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708. IEEE.

International Multiple Sclerosis Genetics Consortium (2010). Evidence for polygenic susceptibility to multiple sclerosis—the shape of things to come. *The American Journal of Human Genetics*, 86(4):621–625.

International Schizophrenia Consortium (2009). Common polygenic variation contributes to risk of schizophrenia and bipolar disorder. *Nature*, 460(7256):748–752.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Khera, A. V., Chaffin, M., Aragam, K. G., Haas, M. E., Roselli, C., Choi, S. H., Natarajan, P., Lander, E. S., Lubitz, S. A., Ellinor, P. T., et al. (2018). Genome-wide polygenic scores for common diseases identify individuals with risk equivalent to monogenic mutations. *Nature Genetics*, 50(9):1219–1224.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krogh, A. and Hertz, J. A. (1992). A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems*, pages 950–957. NIPS.

LeBlanc, M. and Tibshirani, R. (1996). Combining estimates in regression and classification. *Journal of the American Statistical Association*, 91(436):1641–1650.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Li, Q., Cai, W., Wang, X., Zhou, Y., Feng, D. D., and Chen, M. (2014). Medical image classification with convolutional neural network. In *Control Automation Robotics & Vision (ICARCV), 2014 13th International Conference on*, pages 844–848. IEEE.

Locke, A. E., Kahali, B., Berndt, S. I., Justice, A. E., Pers, T. H., Day, F. R., Powell, C., Vedantam, S., Buchkovich, M. L., Yang, J., et al. (2015). Genetic studies of body mass index yield new insights for obesity biology. *Nature*, 518(7538):197–206.

Montañez, C. A. C., Fergus, P., Montañez, A. C., Hussain, A., Al-Jumeily, D., and Chalmers, C. (2018). Deep learning classification of polygenic obesity using genome wide association study snps. *arXiv preprint arXiv:1804.03198*.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814.

Nesterov, Y. (1983). A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$. In *Soviet Mathematics Doklady*, volume 27, pages 372–376.

Ng, A. (2000). Cs229 lecture notes. `http://backspaces.net/temp/ML/CS229.pdf`. Online; Accessed: 2018-05-30.

Ng, A. Y. (2004). Feature selection, L_1 vs. L_2 regularization, and rotational invariance. In *Proceedings of the Twenty-First International Conference on Machine learning*, page 78. ACM.

Paré, G., Mao, S., and Deng, W. Q. (2017). A machine-learning heuristic to improve gene score prediction of polygenic traits. *Scientific Reports*, 7(1):12665.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, pages 1310–1318. PMLR.

Prechelt, L. (1998). Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, 11(4):761–767.

Price, A. L., Patterson, N. J., Plenge, R. M., Weinblatt, M. E., Shadick, N. A., and Reich, D. (2006). Principal components analysis corrects for stratification in genome-wide association studies. *Nature Genetics*, 38(8):904.

Puurula, A., Read, J., and Bifet, A. (2014). Kaggle lshtc4 winning solution. *arXiv preprint arXiv:1405.0546*.

Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151.

Raschka, S. (2015). *Python Machine Learning*. Packt Publishing, Birmingham, UK.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, 22:400–407.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Samek, W., Wiegand, T., and Müller, K.-R. (2017). Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296*.

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Simonson, M. A., Wills, A. G., Keller, M. C., and McQueen, M. B. (2011). Recent methods for polygenic analysis of genome-wide data implicate an important effect of common variants on cardiovascular disease risk. *BMC Medical Genetics*, 12(1):146–155.

Smith, L. N. (2017). Cyclical learning rates for training neural networks. In *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*, pages 464–472. IEEE.

Speliotes, E. K., Willer, C. J., Berndt, S. I., Monda, K. L., Thorleifsson, G., Jackson, A. U., Allen, H. L., Lindgren, C. M., Luan, J., Mägi, R., et al. (2010). Association analyses of 249,796 individuals reveal 18 new loci associated with body mass index. *Nature Genetics*, 42(11):937–948.

Spiliopoulou, A., Nagy, R., Bermingham, M. L., Huffman, J. E., Hayward, C., Vitart, V., Rudan, I., Campbell, H., Wright, A. F., Wilson, J. F., et al. (2015). Genomic prediction of complex human traits: relatedness, trait architecture and predictive meta-models. *Human Molecular Genetics*, 24(14):4167–4182.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Sudlow, C., Gallacher, J., Allen, N., Beral, V., Burton, P., Danesh, J., Downey, P., Elliott, P., Green, J., Landray, M., et al. (2015). UK Biobank: an open access resource for identifying the causes of a wide range of complex diseases of middle and old age. *PLoS medicine*, 12(3):e1001779.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In Dasgupta, S. and McAllester, D., editors, *International Conference on Machine Learning*, pages 1139–1147. JMLR.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude.

Töscher, A., Jahrer, M., and Bell, R. M. (2009). The bigchaos solution to the Netflix grand prize. `https://www.netflixprize.com/assets/GrandPrize2009_BPC_BigChaos.pdf`. Online; Accessed: 2018-09-12.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008. NIPS.

Vilhjálmsson, B. J., Yang, J., Finucane, H. K., Gusev, A., Lindström, S., Ripke, S., Genovese, G., Loh, P.-R., Bhatia, G., Do, R., et al. (2015). Modeling linkage disequilibrium increases accuracy of polygenic risk scores. *The American Journal of Human Genetics*, 97(4):576–592.

Ware, E. B., Schmitz, L. L., Faul, J. D., Gard, A., Mitchell, C., Smith, J. A., Zhao, W., Weir, D., and Kardia, S. L. (2017). Heterogeneity in polygenic scores for common human traits. *bioRxiv: 106062*.

Warren, H., Casas, J.-P., Hingorani, A., Dudbridge, F., and Whittaker, J. (2014). Genetic prediction of quantitative lipid traits: comparing shrinkage models to gene scores. *Genetic Epidemiology*, 38(1):72–83.

Whittaker, J. C., Thompson, R., and Denham, M. C. (2000). Marker-assisted selection using ridge regression. *Genetics Research*, 75(2):249–252.

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., and Recht, B. (2017). The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, pages 4151–4161. NIPS.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, 5(2):241–259.

Wood, A. R., Esko, T., Yang, J., Vedantam, S., Pers, T. H., Gustafsson, S., Chu, A. Y., Estrada, K., Luan, J., Kutalik, Z., et al. (2014). Defining the role of common variation in the genomic and biological architecture of adult human height. *Nature Genetics*, 46(11):1173.

Yang, J., Benyamin, B., McEvoy, B. P., Gordon, S., Henders, A. K., Nyholt, D. R., Madden, P. A., Heath, A. C., Martin, N. G., Montgomery, G. W., et al. (2010). Common SNPs explain a large proportion of the heritability for human height. *Nature Genetics*, 42(7):565.

Yue, T. and Wang, H. (2018). Deep learning for genomics: A concise overview. *arXiv preprint arXiv:1802.00810*.

Zeng, H., Edwards, M. D., Liu, G., and Gifford, D. K. (2016). Convolutional neural network architectures for predicting DNA–protein binding. *Bioinformatics*, 32(12):i121–i127.

Zhou, J. and Troyanskaya, O. G. (2015). Predicting effects of noncoding variants with deep learning–based sequence model. *Nature Methods*, 12(10):931.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320.