Disease Risk Prediction from Genome Sequence Data

Kevin Zhang, Sicheng Jia, Yuheng He

Dec 4, 2021

1 Problem Definition

An individual's DNA represented by a genotype can often be a good indicator of how likely they will be affected by a disease represented by a phenotype. This can be leveraged to build a predictive model that is able to predict an individual's phenotype given only their genotypes. In this paper, we experiment with 2 types of simulation-generated genotype-phenotype disease data: infinitesimal and non-infinitesimal. The infinitesimal data signifies that all SNPs in the genotype are simulated to contribute to the phenotype while the non-infinitesimal data signifies that only one percent of SNPs contribute to the phenotype. We use these datasets to build an accurate predictive model.

2 Challenges

2.1 Dimensionality

In bioinformatic studies, genotype data are frequently used to make analyses, and are often given in the form of sequences of SNPs. Although SNPs only constitute a small fraction of human genome, there are still millions of SNPs to be considered when using the data to make analyses and predictions. On one hand, large scale data allow us to not only more accurately explore the hidden features in the genotype, but also extract more significant common features across different populations even when there are large variations among them. However, the large scale and high dimensionality of SNP data make it extremely computationally expensive to process, and also reveal several other difficulties such as noise accumulation and algorithmic instability. In most real world cases, it is infeasible to use all SNP data in a predictive model while expecting a reasonable time for training completion and efficient prediction.

2.2 Non-linearity

Linkage Disequilibrium (LD) is another phenomenon brings predictive challenges to genomic data. LD manifests in genotypes as non-random associations between alleles at different loci, meaning that nearby SNPs are sometimes associated with each other. Such a linkage may result in nonlinear relationship between SNPs, and thus when we are processing different SNPs, we can no longer assume they are linearly independent. In addition, the effect some SNPs have on the phenotype may not be additive. These two factors contribute to a certain degree of non-linearity in the data and decrease the effectiveness of simple linear models like linear regression.

3 Approaches

3.1 Dimensionality

To tackle the high dimensionality of the data, we can use ***Principle Component Analysis (PCA)*** to perform dimensionality reduction. PCA is a technique for reducing the dimensionality of large scale datasets, increasing interpretability but at the same time minimizing information loss. The mechanism behind PCA is that it attempts to map the input data into lower dimensions such that the variance of the data is maximized in the lower dimension. Principal components are eigenvectors of the data's covariance matrix. Thus, the principal components are often computed by eigendecomposition of the data covariance matrix or singular value decomposition of the data matrix. In order to choose the number of dimensions, we can look at the eigenvalues after running PCA, and choose the dimension that balances between information loss and number of dimensions. [1]

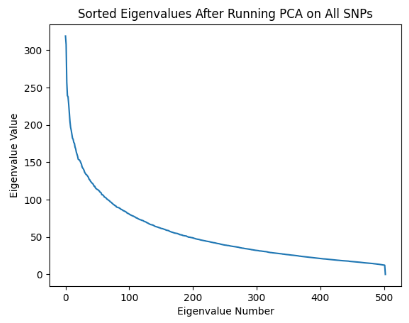

Figure 1: Sorted Eigenvalues After Running PCA on All SNPs

After performing PCA, we find that the percentage of variance explained by the eigenvectors drops significantly at after the 50th eigenvector, indicating that the first 50 components preserve most of the genotype information. To be more specific, we choose to preserve 50 components to implement PCA which retain 65% of the variance, while still keeping the dimension low to prevent complexity explosion.

3.2 Non-linearity

To address the issue of non-linearity between SNPs, we explore more powerful models such as neural network and regularization techniques such as ***Ridge Regression*** and ***Elastic Net*** to compare them against the baseline ***Linear Regression*** model. Linear regression establishes a relationship between phenotype data and genotype data using a best fit regression line. As we mentioned before, linear regression model assumes linearity of the data so the result may not have enough predictive power. Ridge regression reduces the standard errors by adding a degree of bias to the regression estimates. Elastic net combines characteristics of both lasso and ridge regression, reducing the impact of different features while not eliminating all of the features. For both our Ridge Regression and Elastic Net models we empirically find that an α (regularization constant) of 25 is able to perform the best. For the implementation of our linear models, we use the linear model module from scikit-learn [4].

Neural networks are a class of more powerful prediction models that are to learn non-linear functions due to its non-linear activation functions. Although using network minimizes the effect of non-linearity in the data, it comes at the cost of increasing the training time as well as the number of parameters to tune. In our project, we use several neural network architectures such as ***Fully Connected (FC)*** and ***Convolutional Neural Networks (CNNs)***. Regularization techniques such as Dropout are also applied to these networks. For the FC architecture, we test two configurations. The first FC model called 'FC', takes the 50 dimension input and feeds it into 5 fully connected layers with 50, 50, 25, 10, and 1 nodes in each respective layer. The second FC model, called 'FC DROP' has the same configuration as the first, but with dropout layers in between each fully connected layers. For both FC models, a ReLU activation function is applied after each layer. For the CNN architecture, we also test two configurations. The first CNN model, called 'CNN SIMPLE' feeds the input into a 1-dimensional convolution layer with a filter size of 20 and a kernel size of 4, which is then flattened and fed into 3 fully connected layers with 25, 10, and 1 nodes. The second CNN model, called 'CNN' is similar to the first, but contains 3 1-dimensional convolution layers with a 1-dimensional MaxPool layer in between each convolution layer. For both CNN models, Dropout and a ReLU activation function are applied after each layer. For the implementation of the neural networks, we use the layers and optimizers from TensorFlow [3]. Finally, we also test RNN architectures that try to find time-variant or time-dependent variance in the data. We perform training using a neural network with a single LSTM layer with 4 units and a single GRU layer with 4 units.

4 Evaluation

Since the phenotype is a continuous measurement, we can use the ***Mean Squared Error (MSE)*** to evaluate the performance of each of our models, because it results in a convex loss function for linear regression models and closed form gradients for neural network architectures. For the linear model, minimizing the MSE can contribute to the optimization of the model, and as for neural networks, they are trained using an optimization process which requires a loss function to calculate the model error, where MSE is a great candidate function. Furthermore, we use cross-validation to split the testing data and training data. When the same training data and testing data are used for our predictive models, we can compare their performance by comparing the corresponding MSE on the test data.

5 Datasets

We use imputed data from chromosome 22 as the genotype data, which consists of 83,678 SNPs from 503 individuals [5]. The phenotype data given are generated from two types of models. The infinitesimal model (allcausal), which assumes every SNP affect has an effect on the trait, and Non-infinitesimal model (OnePercCausal), where only one percent of SNPs have a non-zero effect on the trait. Additionally, the phenotype data are comprised of 5 different simulations using $h^2 = \{0.0, 0.2, 0.4, 0.8\}$, which refer to the proportion of the variance in the simulated trait that is due to genetics. To create the training and test datasets, we use simulation 1 from each h^2 value and perform a 3:1 training/test split.

6 Results

With the processed data, we split our data into training set and testing set as specified in the previous section. Then, we use the training dataset to train our predictive models, and use the testing dataset to calculate the MSE. The MSE values are plotted such that different h^2 values on the same dataset shown in Fig. 2. A table of all MSE values for both datasets is also given in below.

From the results, we see that for the infinitesimal dataset, because every SNP has a causal effect on the phenotype, the performances across all predictive models are consistent. Models have a better predictive power with higher h^2 because the genotype data has a greater effect on the phenotype. For example, with $h^2 = 0.0$ there is zero variance in the simulated trait, and hence the genotype and phenotype are not connected, which explains its low predictability. With $h^2 = 0.8$, on the other hand, the effective proportion goes up to 80% so the genotype and phenotype much more closely associated, giving us low MSE for the models.

For the non-infinitesimal (one percent) dataset, we see that previous conclusion does not hold because in this dataset, only one percent of the SNPs contribute to the phenotype, leaving the rest of the data acting as noises. Therefore, a higher h^2 value does not guarantee a better predictive outcome due to the interference of the non-causal SNPs. One consistent result we find is that data with $h^2 = 0.0$ always has a higher MSE regardless of which model we implement. This is because of the 0 effective proportion from the

MSE on the Infinitesimal Dataset				
Model	$h^2 = 0.0$	$h^2 = 0.2$	$h^2 = 0.4$	$h^2 = 0.8$
Linear	0.157873	0.127768	0.115064	0.11121
Ridge	0.157853	0.127758	0.115047	0.111199
ElasticNet	0.133297	0.125987	0.115039	0.110114
FC	0.500796	0.47138	0.746204	0.414749
FC DROP	**0.131613**	**0.125559**	0.11561	**0.109985**
CNN SIMPLE	0.133292	0.125993	0.115043	0.110115
CNN	0.133084	0.125977	**0.115017**	0.110097
MSE on the One Percent Causal Dataset				
Model	$h^2 = 0.0$	$h^2 = 0.2$	$h^2 = 0.4$	$h^2 = 0.8$
Linear	0.124645	0.087332	0.116067	0.100247
Ridge	0.124633	0.087308	0.116051	0.100235
ElasticNet	0.113285	**0.065422**	0.106436	**0.099317**
FC	0.701909	0.485462	0.481558	0.2891
FC DROP	0.11375	0.066982	**0.105725**	0.100026
CNN SIMPLE	**0.113236**	0.065435	0.106507	0.099362
CNN	0.113257	0.065434	0.10649	0.099325

Table 1: Mean Squared Errors of Different Models on the Infinitesimal and One Percent Causal Dataset

genotype. Data with $h^2 = 0.2$ have a surprisingly low MSE and we extrapolate that our model find some hidden association in the data, that is unrelated to the non-zero effect genetic composition.

In terms of the performance of each model, we find that the Elastic Net model performs consistently well in all datasets and all of the linear regression models perform consistently well on fitting on the data. We also discover that the Fully Connected Network without regularization performs poorly on the testing dataset; when regularizations such as Dropout and Max Pooling are applied however, the models end up performing very well and yield the lowest MSE on average for the infinitesimal model. We also find that the linear regression model performs similarly across the board which suggests that there is a linear association between the SNPs and the phenotype in the PCA dimension. However, any RNN layer training reveals 0 evidence of any time variance or time-dependent data. This definitely makes sense since we peruse the logs for the generation and there is no evidence that the data is generated with time as a parameter, providing a sanity check that a normal neural network is the correct network to train on.

All in all, we can conclude that h^2 values affect the predictive accuracy, the traits of the dataset also have an impact on the outcome.

6.1 Remarks

We originally found that the MSE of the models on the $h^2 = 0.0$ dataset was much lower than the MSE of the other datasets(an unexpected result due to the fact that we hypothesized heritability to be inversely correlated with MSE). We initially believed this to be due to hidden associations within the data. However,

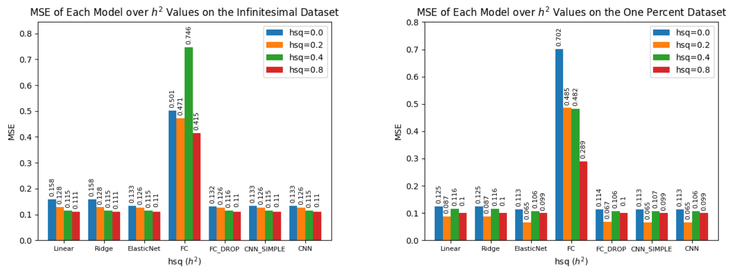

Figure 2: MSE of Different Models over different h^2 Values on the Infinitesimal and One Percent Causal Dataset

after running a permutation test to uncover any associations, we found that there were indeed no associations between the genotype and phenotype data. This led us to further investigate the dataset, after which we found that the phenotype values were not normalized between datasets. After normalizing the phenotype values and rerunning each model, we obtained the values described above which aligned with previous findings.

7 Continuation

For future study, we can try refining the dimensionality reduction techniques using more complex methods such as Kernel PCA, or different ones such as Factor Analysis (FA). Furthermore, we can combine the dimensionality reduction process with the training process, for example, applying the CNNs within small windows and then combining predictions across windows gives us a combination of both steps. We can also use Genetic Algorithm to train our models in order to find the parameters that yield optimal results.

Moreover, the hyperparameters of our convolutional neural networks are far from fine-tuned. Given more time, we can definitely fine tune our convolutional neural networks by tuning the sgd batch size, the actual epochs and learning rate, and also the actual layers by selecting more targeted feature sets for training. Furthermore, there are many papers with deep neural networks proven to learn super-linear training data including AlexNet, VGGNet, and other complex neural networks that we can fit onto our data and perhaps find inspiration from in creating our own neural networks[2].

8 Conclusion

We use PCA to reduce the dimensionality of our data, which allows us to create both linear and non-linear predictive models that map an individual's genotype to their phenotype. We find that as the h^2 heritability increases, the MSE of our models decreases which aligns with our original understanding of how heritability works(more heritability results in a higher level of predictiveness based on our given genotype data). We also find that Elastic Net, FC DROP, and the two CNN models are able to achieve relatively performance results, resulting in similarly low mean squared errors. When taking into account the large number of parameters to tune and the long training times, neural networks may not be favored over Elastic Net as a predictive model, as elastic net is sufficient in both accuracy and minimal complexity for effective phenotype prediction. Note that this does not mean that we should discount neural network models for all phenotype predictions. The neural networks have much more flexibility in finding hidden LD associations that cannot be explained by a linear model. We may need to perform more hyper-parameter tuning to find the correct features that will reveal these hidden features.

References

[1] Ian T. Jolliffe and Jorge Cadima. “Principal component analysis: a review and recent developments”. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374.2065 (Apr. 2016), p. 20150202. DOI: `10.1098/rsta.2015.0202`. URL: `https://doi.org/10.1098/rsta.2015.0202`.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. NIPS’12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 1097–1105.

[3] Martın Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[4] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[5] Sriram Sankararaman. *Genotype-Phenotype Data for Disease Risk*. `https://drive.google.com/file/d/1IqRqXFbpQBxinmCcBOQVk_7tUCJbzp5B/view?usp=drive_web`. Accessed: 2021-11-15.