Secure Genotype Imputation using HEAAN

Duhyeong Kim[1], Wonhee Cho[1], Yongha Son[1], Seungwan Hong[1], Jung Hee Cheon[1 2]

[1] Seoul National University

[2] Cryptolab

Objectives

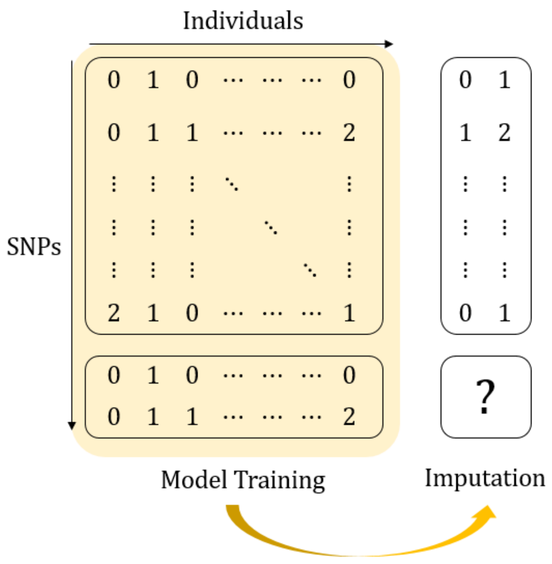
- Brief Overview of Genotype Imputation
- Data Encoding
- Our DNN Configuration
- Data Windowing Technique
- Experimental Results

Brief Overview of Genotype Imputation

Genotype imputation is used to predict missing and low-quality variant genotypes in large scale genotyping projects. Genotype imputation is usually performed on single nucleotide polymorphisms (SNPs).

Matrix Representation of Genotype Imputation

Predicting missing SNPs can be thought as filling in the missing part of the matrix, based on yellow-lighted data.

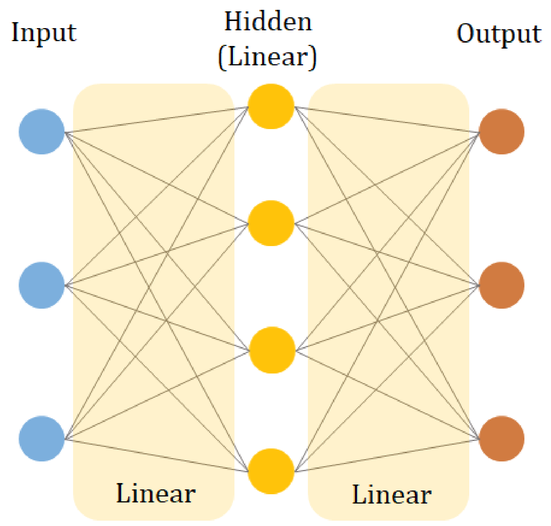


Data Encoding

Although the original `0`, `1`, `2` data can be fed into DNN training without encoding, we encode them into data into `001`, `010`, `100` respectively with a belief that this is more suitable for our purpose.

Moreover, we treat the empty data in tag-SNPs by a zero-data `000`.

Our DNN Configuration

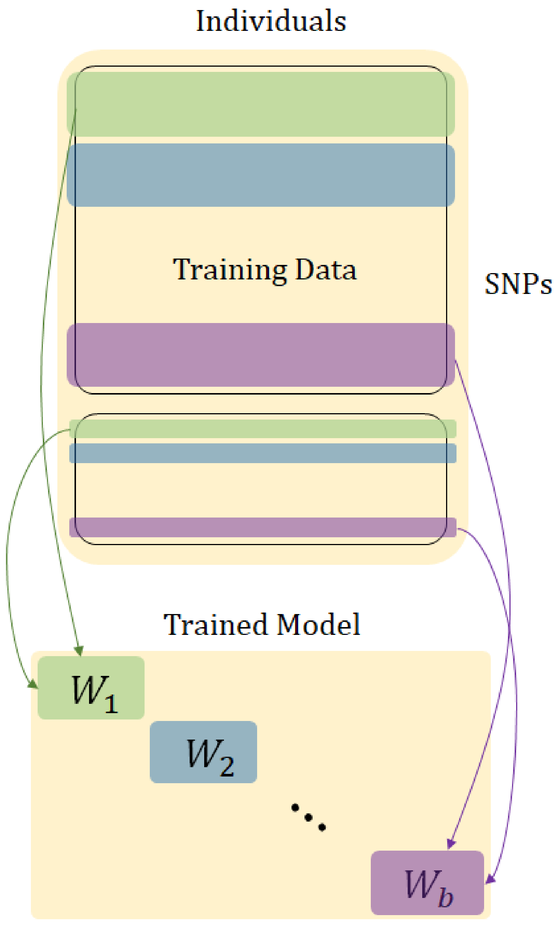


We set one hidden layer with a linear activation function, and output layer with a sigmoid function. Thus our DNN prediction can be done by one matrix multiplication followed by sigmoid evaluation. For the purpose of retaining scores, the last sigmoid evaluation can be omitted and hence *our HE solution only performs one matrix multiplication.*

Remark Our choice of linear activation function is not only for computational optimization. Other activation functions (e.g. ReLU) show worse performance.

Data Windowing Technique

Based on an intuition that a SNP is strongly related to (physically) close SNPs, we divide the train SNP and target SNP into several blocks according to their location data. Therefore, we train models on each divided block to establish a block-diagonal matrix model.



Remark This technique also significantly drops computational cost, as well as the model size.

Experimental Results

We provide experimental results. For encrypted computation, we use HEAAN scheme. Our implementation is done by Linux with Intel Xeon CPU E5-2620 v4 at 2.10GHz processor with 8 threads.

Table: Experimental Results

Exp	Enc (sec)	Comp (sec)	Dec (sec)	Correctness via micro-AUC 0	1	2	total
Exp A	3.8	47.6	2.5	0.999	0.996	0.999	0.998
Exp B	3.8	22.7	2.2	0.946	0.946	0.989	0.977
Exp C	-	-	-	0.978	0.808	0.979	0.943
Exp D	-	-	-	0.939	0.917	0.936	0.929
Exp E	-	-	-	0.754	0.774	0.774	0.757

* Exp C, D, E are performed without encryption

- **Exp A**: 1k SNPs experiment
- **Exp B**: 10k SNPs experiment
- **Exp C**: 10k SNPs experiment without one-hot-encoding
- **Exp D**: 10k SNPs experiment with ReLU
- **Exp E**: 10k SNPs experiment without windowing

Dataset Information

- The number of individuals = 2504
- The number of train SNPs = 1045(10k), 9746(1k)
- The number of target SNPs = 500

Our parameter Selection

1. Algorithm Parameter
   - The number of blocks $b = 38$(10k), 101(1k)
2. HEAAN Parameters (security parameter $\lambda = 128$)
   - $\log q = 54$ (q is the largest modulus)
   - $\log N = 11$ (N is the RLWE dimension)
   - $\log p = 20$ (p is a scaling factor)