Mendel-Impute

Eric C. Chi Hua Zhou Gary K. Chen Diego Ortega Del Vecchyo
Kenneth Lange

Summary

The Mendel-Impute program performs genotype imputation by solving matrix completion problems [3, 7, 2] over a sliding window of SNPs. Specifically window of p SNPs for n individuals is represented as a matrix $\mathbf{X}$ of reference allele dosages. Thus, the element $x_{ij} \in \{0, 1, 2, \text{missing}\}$ denotes the dosage of the reference allele at the jth SNP for the ith individual.

Matrix completion aims to recover an entire matrix when only a small portion of its entries are actually observed. In the spirit of Occam's razor, it seeks the simplest matrix consistent with the observed entries. The intuition behind performing matrix completion over windows of SNPs is to exploit correlation among the SNPs. Namely, matrix completion looks for linkage-disequilibrium structure in the matrices $\mathbf{X}$ over a narrow genomic region. In practice, only a handful of haplotypes occur within a given population over a short region. Accordingly, we expect each $\mathbf{X}$ to have low rank. One way to impute missing genotypes is to find a low-rank matrix that approximates $\mathbf{X}$ well. This suggests the optimization problem

$$\min_{\text{rank}(\mathbf{Z}) \leq r} f(\mathbf{Z}) \quad = \quad \frac{1}{2}\|P_\Omega(\mathbf{X}) - P_\Omega(\mathbf{Z})\|_{\text{F}}^2, \tag{1}$$

where $\|\mathbf{Y}\|_{\text{F}} = (\sum_{i,j} y_{ij}^2)^{1/2}$ denotes the Frobenius norm of a matrix $\mathbf{Y} = (y_{ij})$, Ω indexes the entries that are observed, and $P_\Omega(\mathbf{Y})$ is the projection operator

$$P_\Omega(\mathbf{Y})_{ij} \quad = \quad \begin{cases} y_{ij} & \text{if } (i,j) \in \Omega, \\ 0 & \text{otherwise.} \end{cases}$$

Besides imputing missing entries in $\mathbf{X}$, the optimal $\mathbf{Z}$ also effectively resolves inconsistent genotypes among different platforms. Unfortunately the optimization problem (1) is nonconvex and beset by local minima. Instead we solve the convex relaxation

$$\min \; f(\mathbf{Z}) + \lambda\|\mathbf{Z}\|_*,$$

where the nuclear norm $\|\mathbf{Z}\|_* = \sum_i \sigma_i(\mathbf{Z})$ (sum of the singular values of $\mathbf{Z}$) serves as a surrogate for the rank function rank($\mathbf{Z}$), and λ is a positive parameter that tunes the tradeoff between model fit and model complexity. Mendel-Impute solves problem (2) via the Nesterov method [1].

The Matlab code

The Functions directory contains all the relevant Matlab files. The Matlab function **Mendel_IMPUTE.m** has two required inputs.

1. An input file of a p-by-n matrix of dosages where p is the number of SNPs and n is the number of subjects. 'filename' is read top-to-bottom. The entries in the file should be coded as {0,1,2, NaN}, i.e. a dosage model with respect to a reference allele where NaN indicates a missing entry.
2. A window size w. Three contiguous panels of w SNPs are used to impute the middle window of w SNPs. The next three contiguous panels are obtained by moving the sliding window over by w SNPs. Thus, a subsequent sliding window of SNPs has $2w$ SNPs in common with the previous sliding window.

Thus, to run **Mendel_IMPUTE** type at the MATLAB prompt:

```
>> Z = Mendel_IMPUTE(filename, w)
```

The output is an imputed matrix $\mathbf{Z}$ that results from imputing the middle third of a sliding window of width w. Note that $\mathbf{Z}$ takes on real values, although typically the entries are not much smaller than 0 or much bigger than 2 since the input dosages are in $[0, 2]$. Users will most likely wish to map the entries of $\mathbf{Z}$ to the $[0, 2]$ interval or to an element in $\{0, 1, 2\}$.

A Pedigree Example

- The Pedigree_Example directory contains example files to perform imputation on a synthetic pedigree data set. The input file is **Ped8c_masked_matlab**.
- The Matlab script **demo_Ped8c.m** performs imputation on a random 1% masked set.
- The Matlab script **convertDosage2Genotypes.m** converts the real valued Mendel_IMPUTE values back to standard genotypes using the reference files **alleles_major.txt** and **alleles_minor.txt**.
- For comparison the same data in **Ped8c_masked_matlab** is provided in the Merlin format needed to execute MACH [4, 5, 6].
  - **Ped8c_geno_chr22_jpt+chb.unr_merlin.dat**
  - **Ped8c_geno_chr22_jpt+chb.unr_merlin.ped**
- The perl script **MaskGenotypes.pl** masks entries in the pedigree file according to an input file of masking coordinates. To run the script in this example type the following.

  ```
  perl MaskGenotypes.pl Ped8c_geno_chr22_jpt+chb.unr_validation_coordinates_replicate_1.csv
     Ped8c_geno_chr22_jpt+chb.unr_merlin.ped Ped8c_geno_chr22_jpt+chb.unr_merlin_masked.ped
  ```

- In general, masking can be accomplished with

  ```
  perl MaskGenotypes.pl masking_coordinates input_pedigree.ped output_pedigree.ped
  ```

Reference haplotypes can be applied by including reference individuals who are homozygous for each reference haplotype to the study panel.

References

[1] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sci.*, 2(1):183–202, 2009.

[2] Jian-Feng Cai, Emmanuel J. Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM J. Optim.*, 20(4):1956–1982, 2010.

[3] Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Found. Comput. Math.*, 9(6):717–772, 2009.

[4] Y. Li and G.R. Abecasis. MACH 1.0: rapid haplotype reconstruction and missing genotype inference. *Am J Hum Genet*, S79:2290, 2006.

[5] Yun Li, Cristen Willer, Serena Sanna, and Gonçalo Abecasis. Genotype imputation. *Annual Review of Genomics and Human Genetics*, 10(1):387–406, 2009. PMID: 19715440.

[6] Yun Li, Cristen J. Willer, Jun Ding, Paul Scheet, and Gonçalo R. Abecasis. Mach: using sequence and genotype data to estimate haplotypes and unobserved genotypes. *Genetic Epidemiology*, 34(8):816–834, 2010.

[7] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research*, 11:2287–2322, 2010.