Fast Imputation and Haplotyping

Kenneth Lange

Departments of Biomathematics, Human Genetics, and Statistics
University of California, Los Angeles

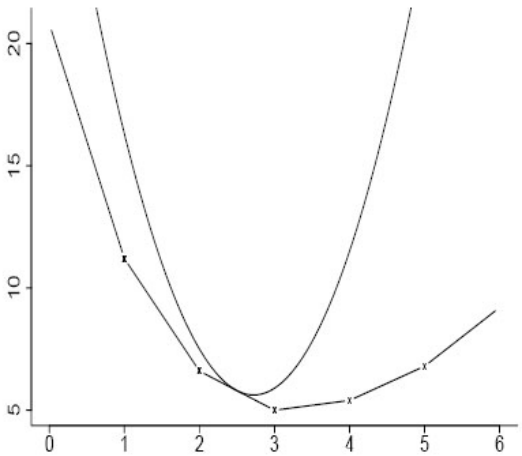
joint work with Janet Sinsheimer, Eric Sobel, Rory Wasiolek, Hua Zhou

Banff, August, 2018

Introduction to the MM Principle

1. The MM principle is not an algorithm, but a prescription or principle for constructing optimization algorithms.
2. The EM algorithm from statistics is a special case.
3. An MM algorithm operates by creating a surrogate function that minorizes or majorizes the objective function. When the surrogate function is optimized, the objective function is driven uphill or downhill as needed.
4. In minimization MM stands for majorize/minimize, and in maximization MM stands for minorize/maximize.
5. MM algorithms are particularly appealing for high-dimensional problems.

Majorization and Definition of the Algorithm



- A function $g(\theta \mid \theta_k)$ is said to majorize a function $f(\theta)$ at θ_k provided

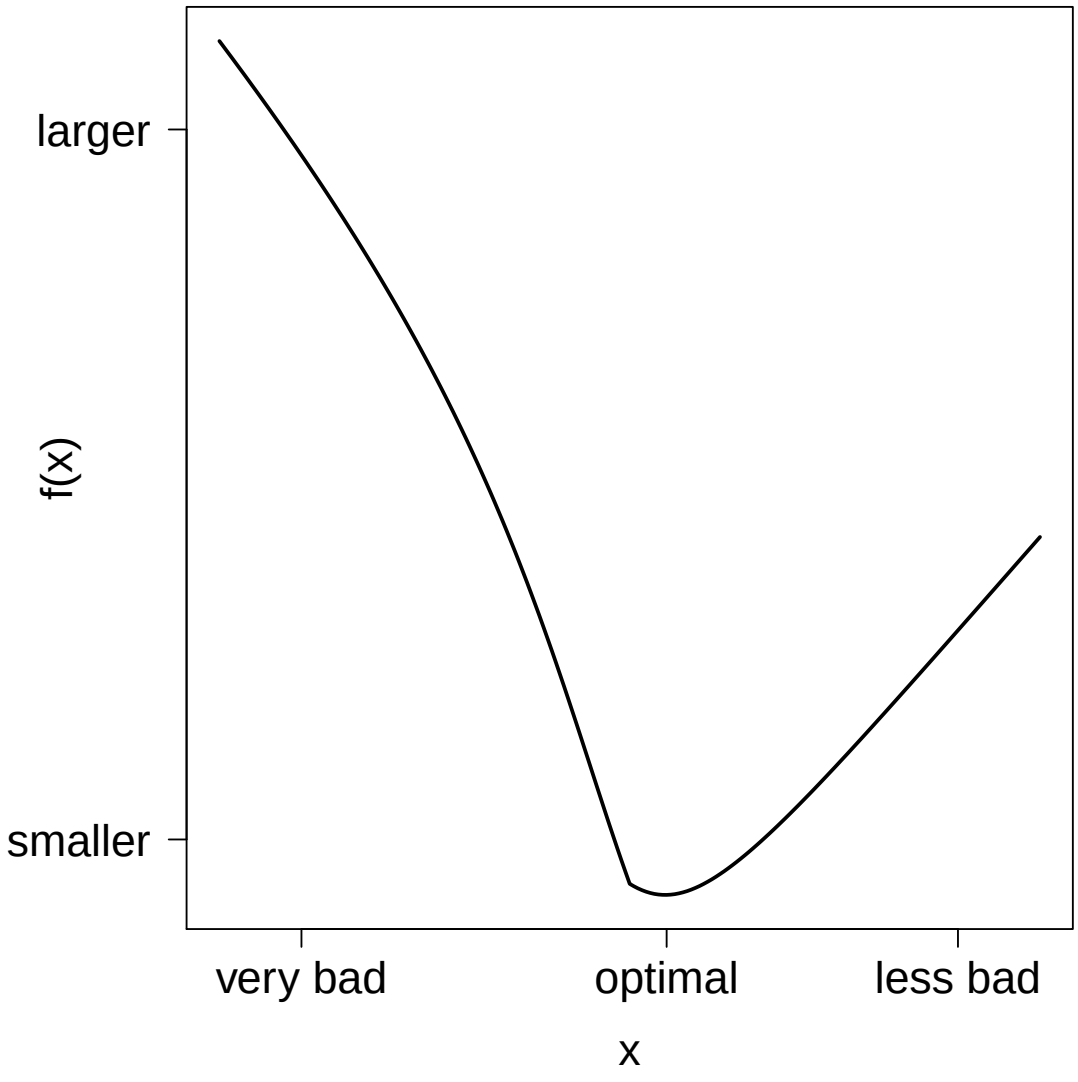
$$f(\theta_k) = g(\theta_k \mid \theta_k)$$
$$f(\theta) \leq g(\theta \mid \theta_k) \quad \text{for all } \theta.$$

- To minimize $f(\theta)$, choose a good majorizing function $g(\theta \mid \theta_k)$ and reduce (usually minimize) it. This produces the next point θ_{k+1} in the algorithm.
- The descent property follows from

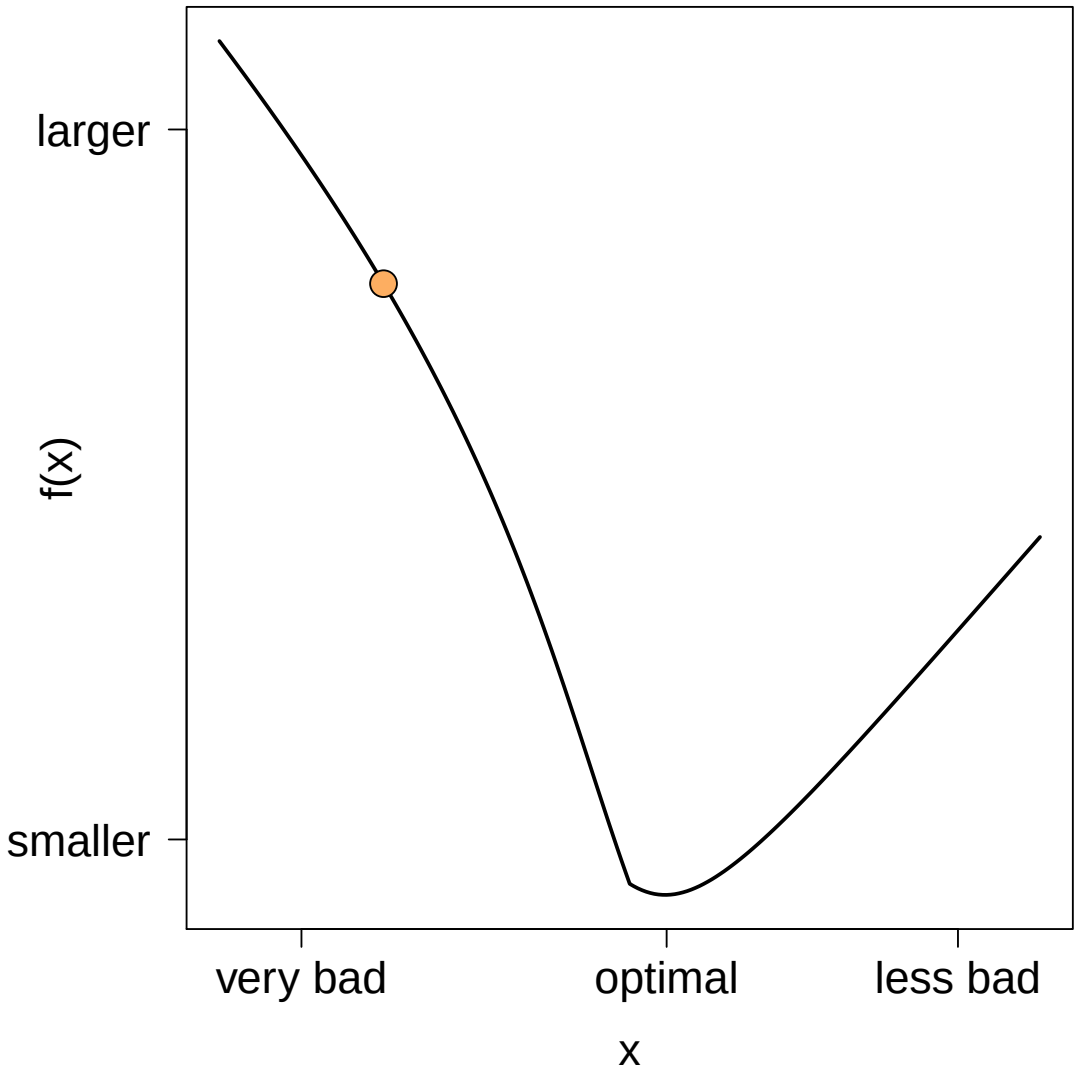
$$f(\theta_{k+1}) \leq g(\theta_{k+1} \mid \theta_k) \leq g(\theta_k \mid \theta_k) = f(\theta_k).$$

The descent property makes MM algorithms very stable.

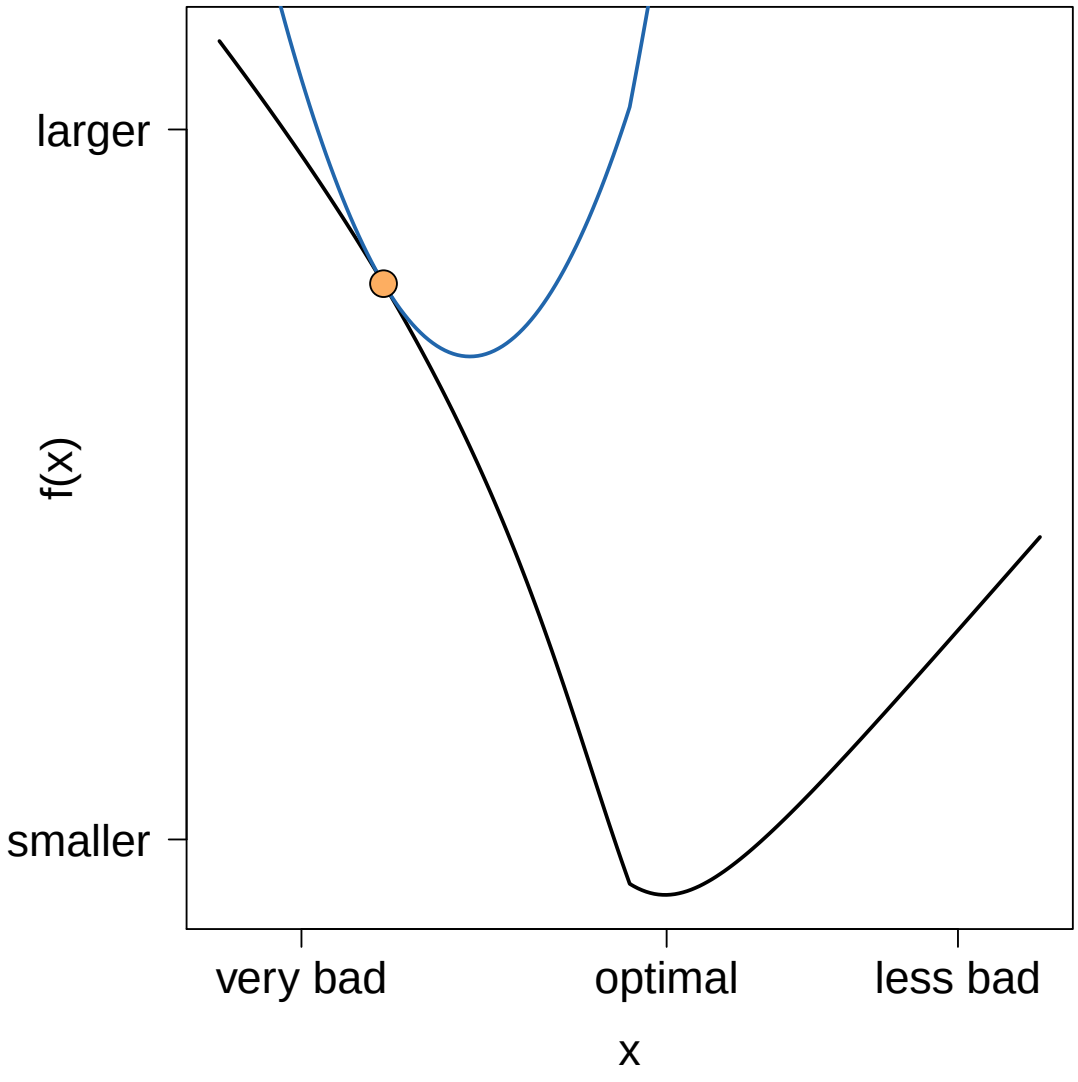
MM Algorithm in Action



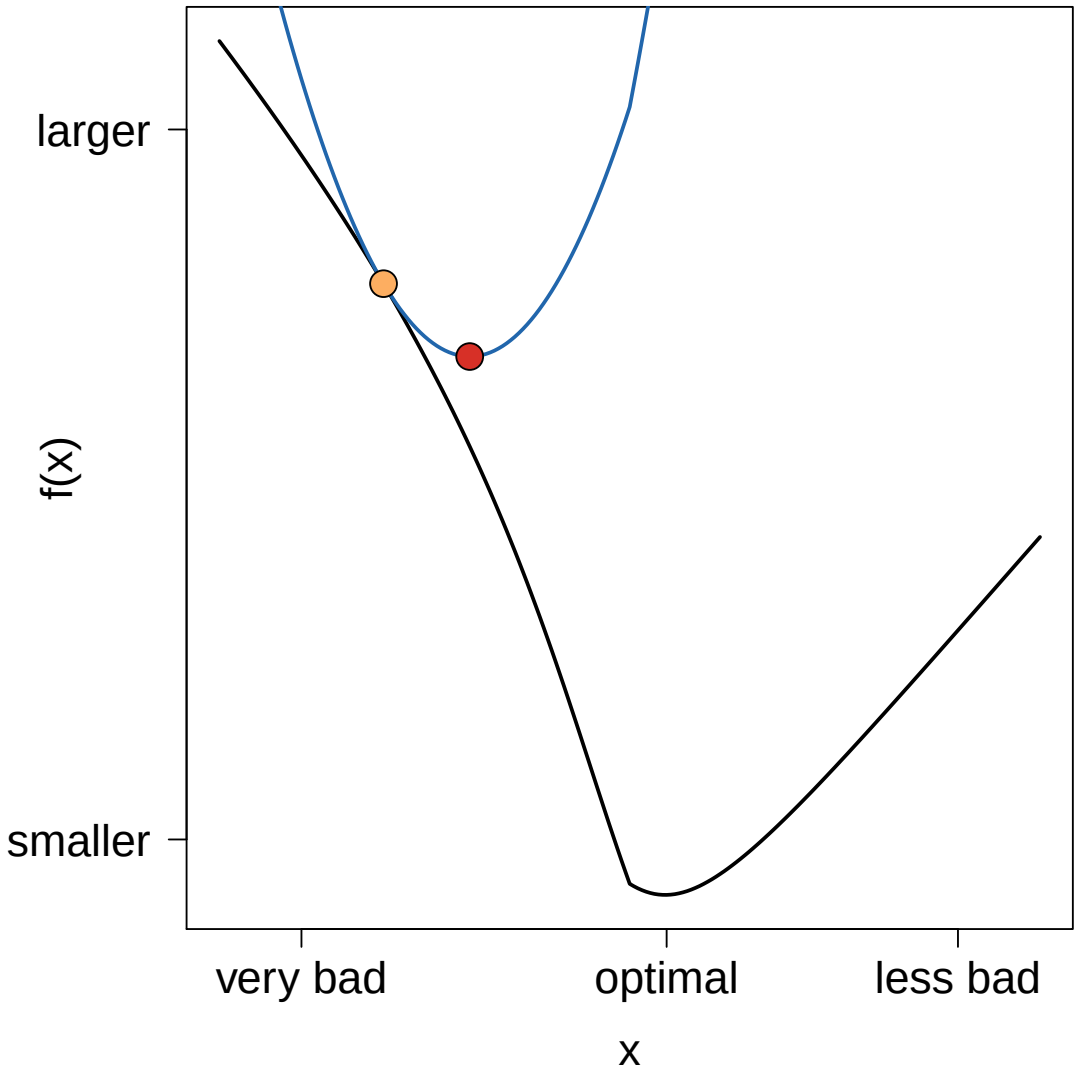
MM Algorithm in Action



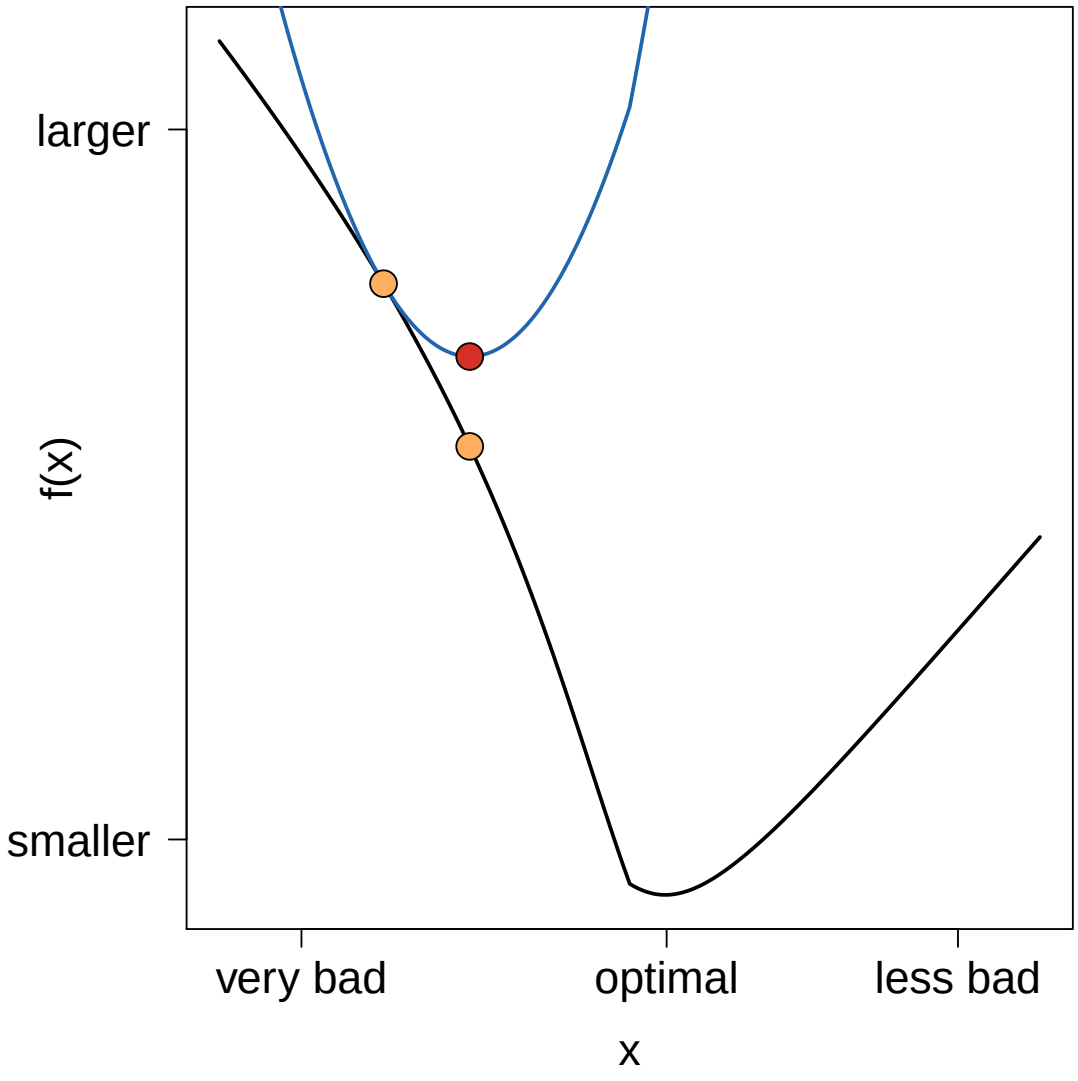
MM Algorithm in Action



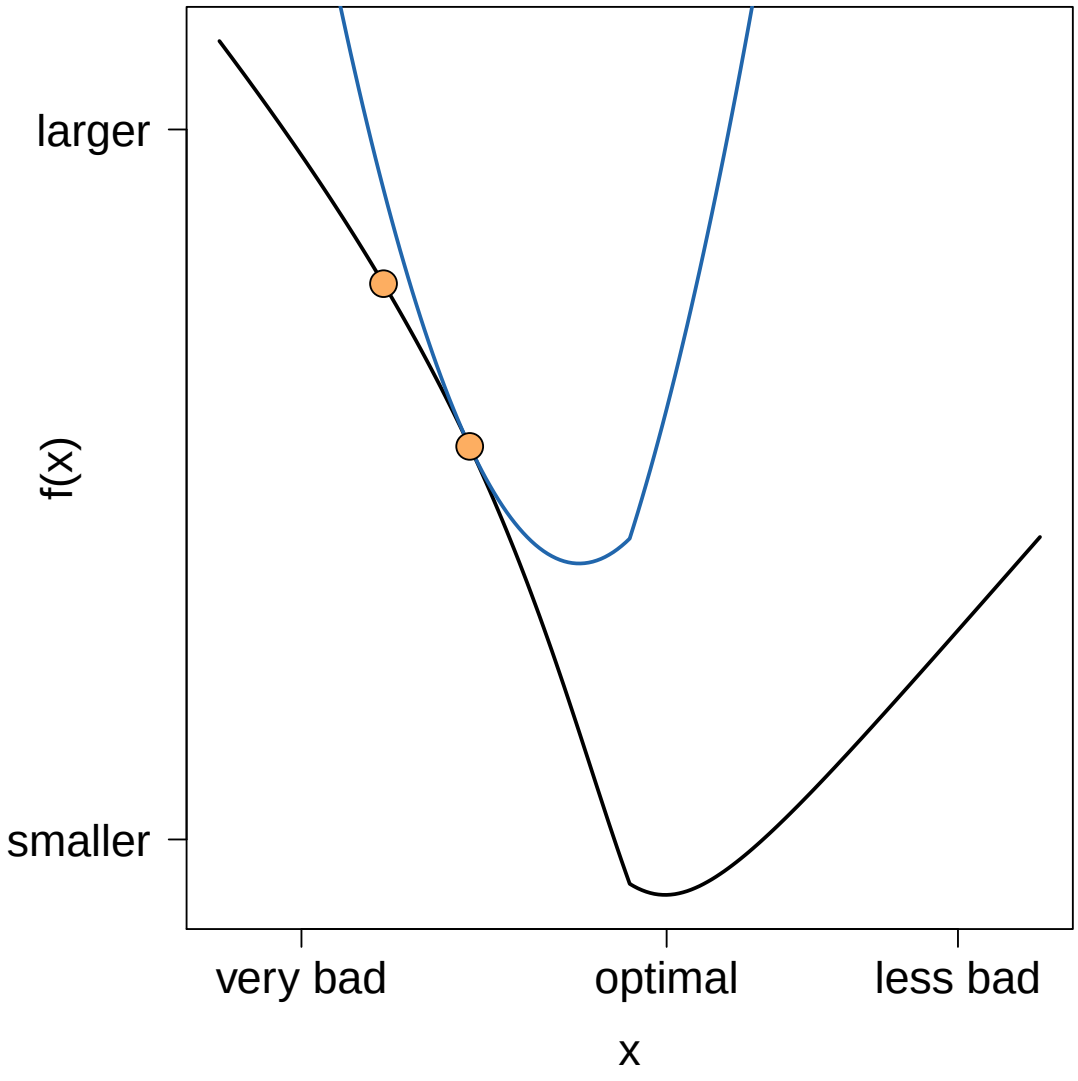
MM Algorithm in Action



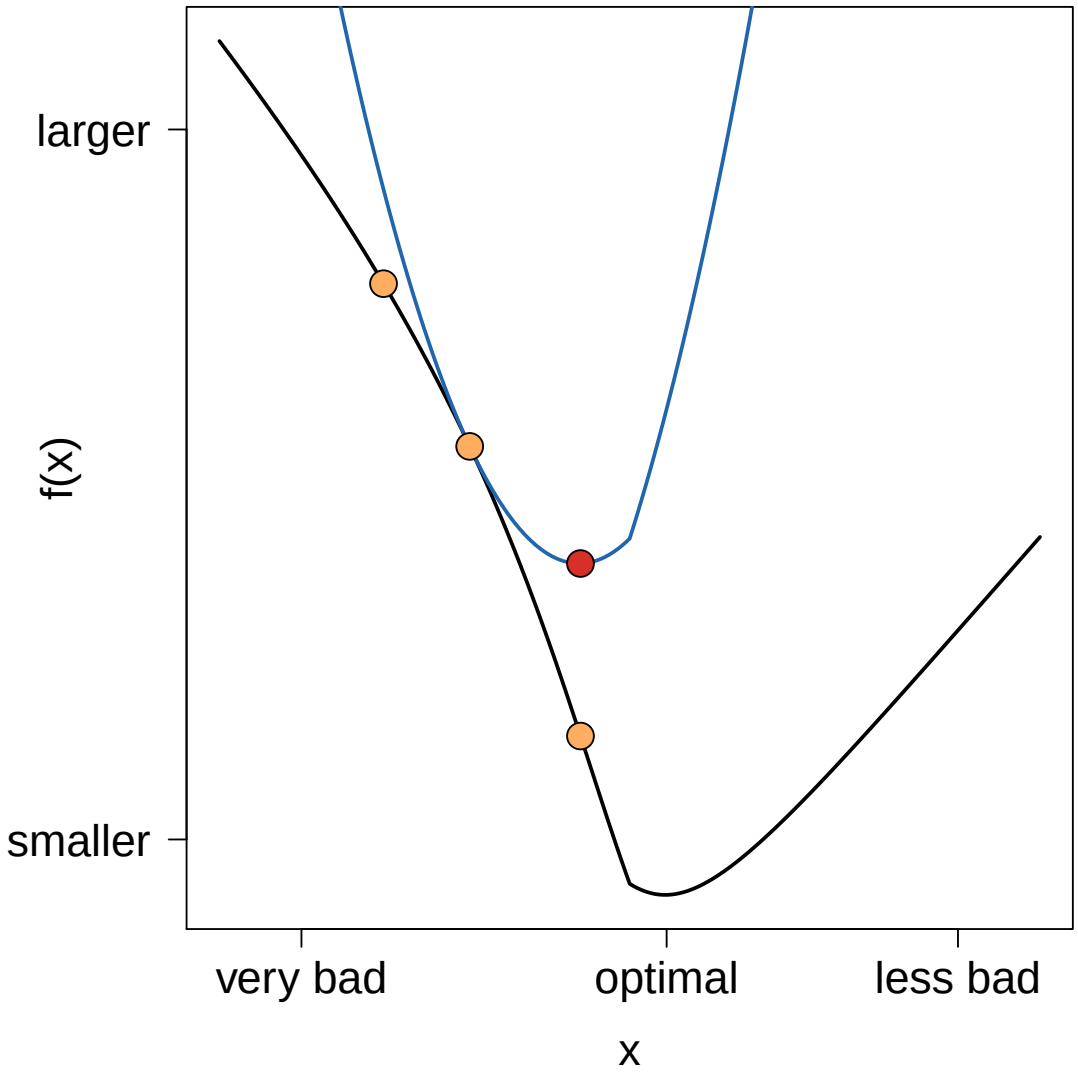
MM Algorithm in Action



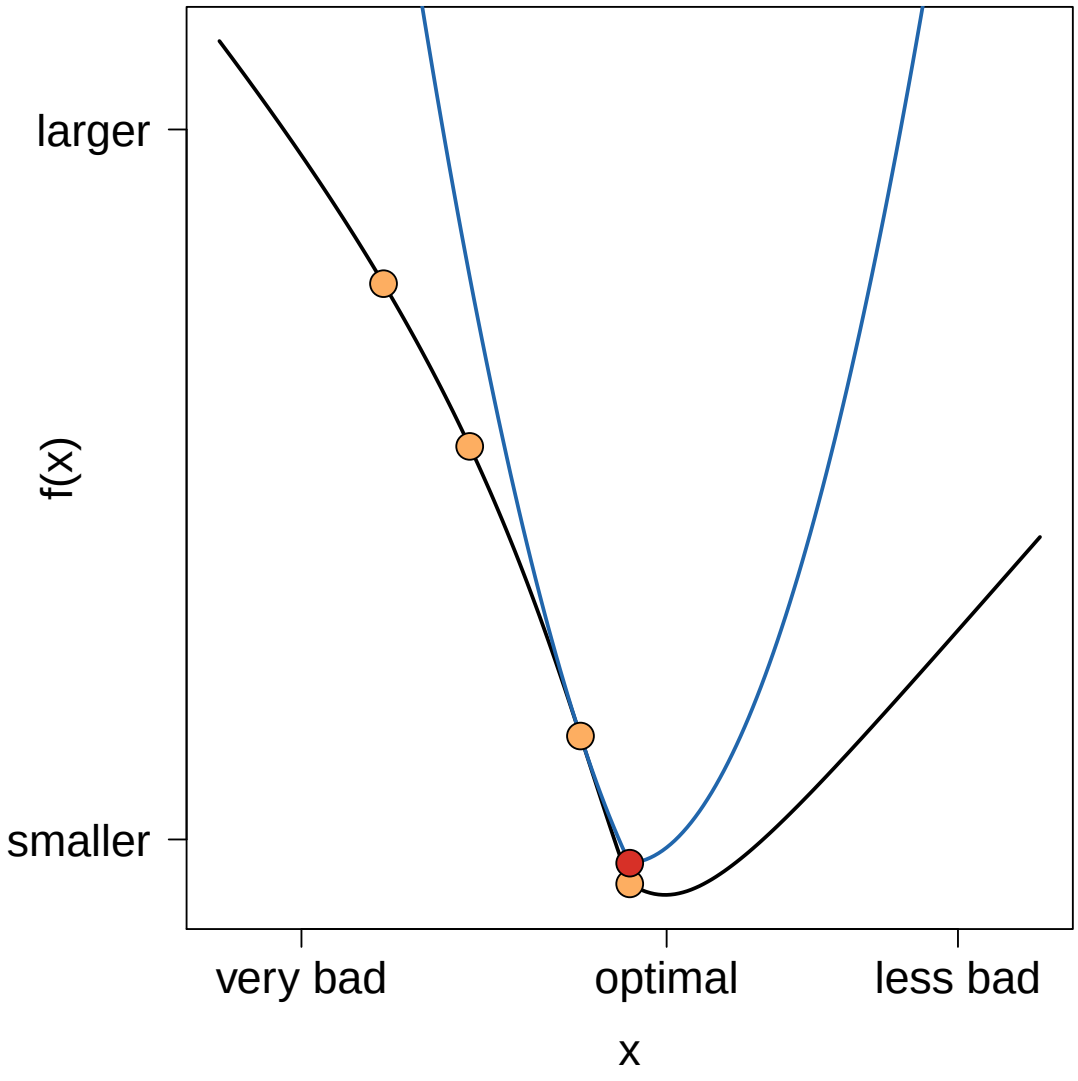
MM Algorithm in Action



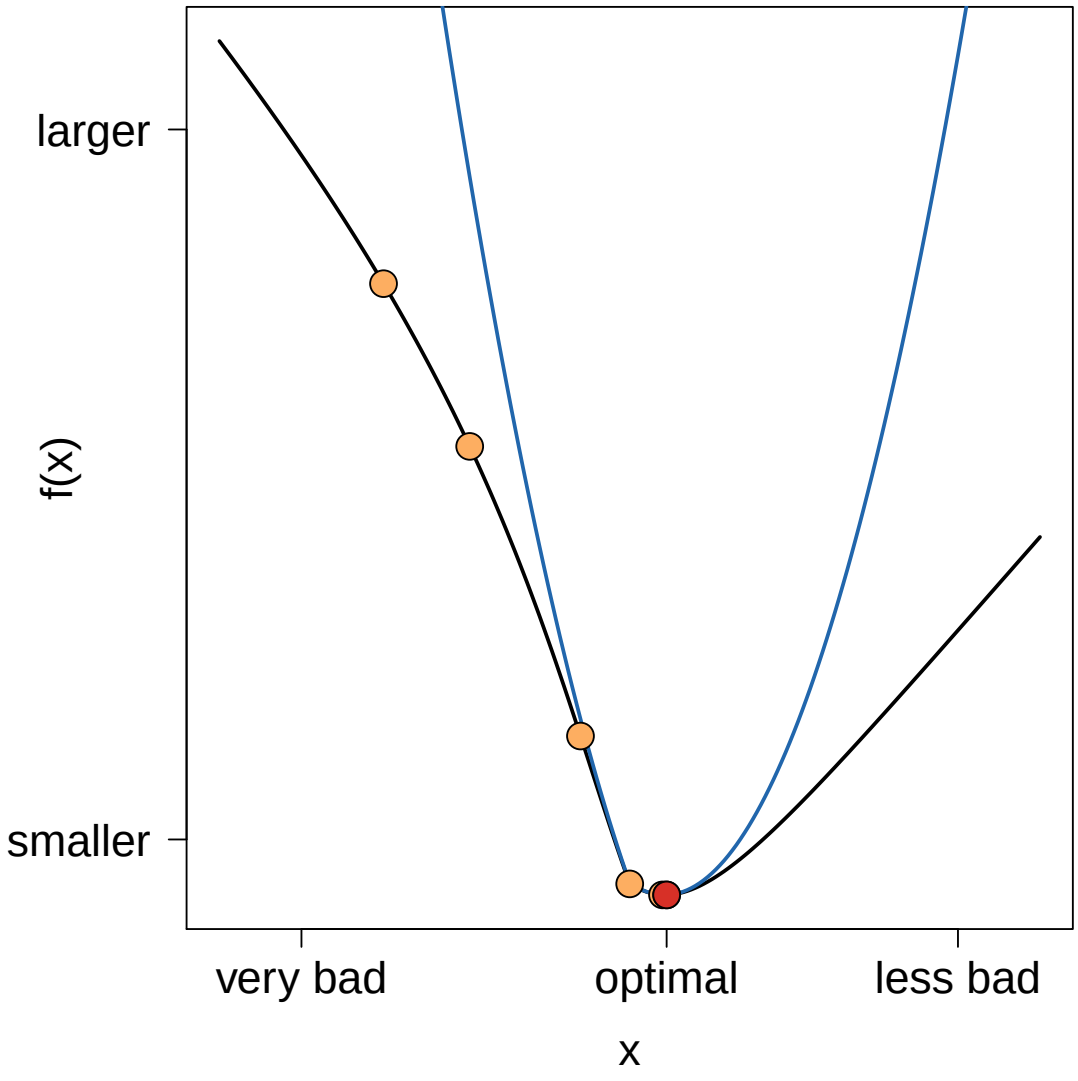
MM Algorithm in Action



MM Algorithm in Action



MM Algorithm in Action



Matrix Completion

- Problem: Given an observed $m \times n$ matrix $X = (x_{ij})$ with observed entries indexed by $\Omega \subset \{1, \ldots, m\} \times \{1, \ldots, n\}$, fill in the missing entries.
- Solution: Find a low rank matrix $\boldsymbol{Z} = (z_{ij})$, consistent with the observed entries of $X = (x_{ij})$.

$$\min_{\text{rank}(Z) \leq r} \sum_{(i,j) \in \Omega} (x_{ij} - z_{ij})^2$$

- The famous Netflix problem fills in missing movie ratings (on a scale of 1 to 5) for moviegoers. The rows of X represent people and the columns movies. Most of the entries are missing.
- Majorize the objective function at iteration k by adding the terms $\frac{1}{2}(z_{kij} - z_{ij})^2$ for missing cells $(i, j) \in \Omega^c$, where $\boldsymbol{Z}_k = (z_{kij})$ equals the current estimate of $\boldsymbol{Z}$.

Objective and Surrogate Functions in Matrix Completion

$$
\begin{aligned}
f(\boldsymbol{U}, \boldsymbol{V}) &= \sum_{(i,j)\in\Omega} \left(x_{ij} - \sum_{n} u_{in} v_{nj} \right)^2 \\
g(\boldsymbol{U}, \boldsymbol{V} \mid \boldsymbol{U}_k, \boldsymbol{V}_k) &= f(\boldsymbol{U}, \boldsymbol{V}) + \sum_{(i,j)\notin\Omega} \left(\sum_{n} u_{kin} v_{knj} - \sum_{n} u_{in} v_{nj} \right)^2
\end{aligned}
$$

Note that

$$
\begin{aligned}
f(\boldsymbol{U}, \boldsymbol{V}) &\geq g(\boldsymbol{U}, \boldsymbol{V} \mid \boldsymbol{U}_k, \boldsymbol{V}_k) \\
f(\boldsymbol{U}_k, \boldsymbol{V}_k) &= g(\boldsymbol{U}_k, \boldsymbol{V}_k \mid \boldsymbol{U}_k, \boldsymbol{V}_k).
\end{aligned}
$$

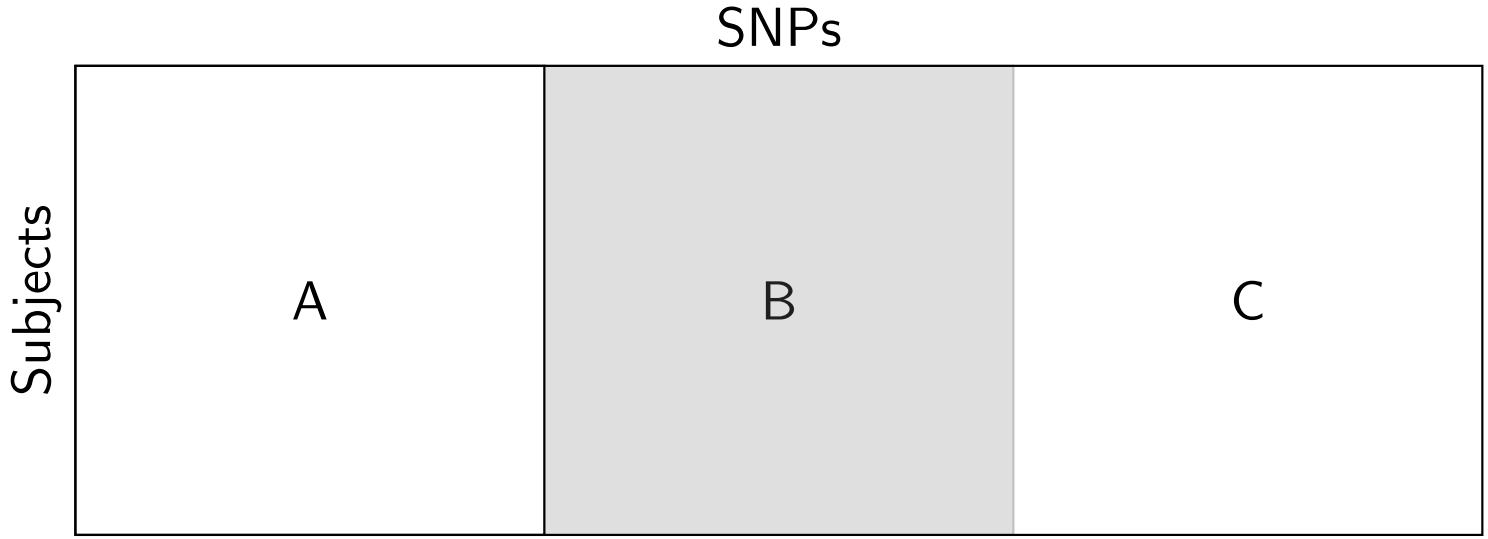
MM Algorithm for a Rank r Approximation

1. Every rank r matrix $\boldsymbol{X}$ of dimension $m \times n$ can be expressed as a matrix product $\boldsymbol{UV}$, where $\boldsymbol{U}$ is $m \times r$ and $\boldsymbol{V}$ is $r \times n$.
2. Each iteration k of the matrix completion MM algorithm alternates between improving $\boldsymbol{U}$ and $\boldsymbol{V}$.
3. For the $\boldsymbol{V}$ update, impute a missing entry x_{ij} of $\boldsymbol{X}$ by $(\boldsymbol{U}_k\boldsymbol{V}_k)_{ij}$. Call the imputed matrix $\boldsymbol{Y}_k$. Minimize the sum of squares $\|\boldsymbol{Y}_k - \boldsymbol{U}_k\boldsymbol{V}\|_F^2$ by the usual regression formula (1)

$$\begin{aligned} \boldsymbol{V}_{k+1} &= (\boldsymbol{U}_k^t\boldsymbol{U}_k)^{-1}\boldsymbol{U}_k^t\boldsymbol{Y}_k && (1) \\ \boldsymbol{U}_{k+1} &= \boldsymbol{Z}_k\boldsymbol{V}_{k+1}^t(\boldsymbol{V}_{k+1}\boldsymbol{V}_{k+1}^t)^{-1}. && (2) \end{aligned}$$

   For the $\boldsymbol{U}$ update, impute a missing entry x_{ij} of $\boldsymbol{X}$ by $(\boldsymbol{U}_{k+1}\boldsymbol{V}_k)_{ij}$. Call the imputed matrix $\boldsymbol{Z}_k$. Minimize the sum of squares $\|\boldsymbol{Z}_k - \boldsymbol{U}\boldsymbol{V}_{k+1}\|_F^2$ by the transposed regression formula (2).
4. The matrices $\boldsymbol{U}_k^t\boldsymbol{U}_k$ and $\boldsymbol{V}_{k+1}\boldsymbol{V}_{k+1}^t$ are $r \times r$ and hence quick to invert when r is small.

Sliding Window (Matrix) of SNP Genotypes



1. Code SNP genotypes as 0, 1, or 2 or as dosages in $[0, 2]$. Exploit linkage disequilibrium to impute in small 3-part windows.
2. Construct a hold-out-set by masking entries in A and C.
3. Train on observed entries from A, B, and C.
4. Choose the matrix rank r based on performance on the hold-out-set.
5. Impute missing entries in the middle third B.

Performance of Matrix Completion on HapMap Data

	CHB					YRI			
	Error (%)		Time (min)			Error (%)		Time (min)	
Chr	MA	MC	MA	MC	Chr	MA	MC	MA	MC
4	4.02	2.15	474	26	5	6.55	2.13	1702	52
5	3.75	2.15	482	27	8	6.38	2.04	1497	45
18	4.28	2.37	296	13	14	6.89	2.36	1173	26
21	4.49	2.51	193	6	15	7.79	2.87	768	22

Table: Accuracy and timing results for MACH (MA) and matrix completion (MC) on eight different chromosomes from the 139 Han Chinese (CHB) and 209 Nigerians (YRI) from HapMap. About 50% of all genotypes are missing. **Caveats**: This is an old comparison. MACH has been upgraded. Error rates diminish with more subjects. Matrix completion timings should improve when alternating least squares is substituted for singular value decompositions.

Extension to Low Coverage Sequencing

- A weighted version of matrix completion applies to sequencing data. One now seeks

$$\min_{\text{rank}(Z)\leq r} \frac{1}{2} \sum_{(i,j)\in\Omega} w_{ij}(x_{ij} - z_{ij})^2,$$

  where w_{ij} is the number of reads at SNP j for subject i and $x_{ij} \in [0, 2]$ is the posterior mean allele dosage. Posterior dosages are derived from Bayes' rule by assuming a binomial likelihood with success equated to a sequencing error and a prior dictated by the Hardy-Weinberg law.
- The minimum can be found by a combination of MM majorization and alternating weighted least squares.

Fast Imputation of Haplotypes

Genotype imputation produces for each person a dosage vector $\boldsymbol{x}$. Phase is still unknown. To recover phase, the current best practice is to exploit reference panels of known haplotypes. The next few slides will discuss a new algorithm for haplotyping. The algorithm operates on sliding windows of SNPs. It then stitches together the inferences from successive windows. The algorithm is fast because it relies on highly parallelized linear algebra.

Imputation within a Window

In an imputation window, let the columns $\boldsymbol{h}_i$ of the matrix $\boldsymbol{H}$ constitute a complete list of reference haplotypes with **redundancies removed**. Haplotype imputation is done by minimizing the criterion

$$\frac{1}{2}\|\boldsymbol{x}-\boldsymbol{h}_1-\boldsymbol{h}_2\|^2=\frac{1}{2}\|\boldsymbol{x}\|^2+\frac{1}{2}\|\boldsymbol{h}_1\|^2+\frac{1}{2}\|\boldsymbol{h}_2\|^2+\boldsymbol{h}_1^t\boldsymbol{h}_2-\boldsymbol{h}_1^t\boldsymbol{x}-\boldsymbol{h}_2^t\boldsymbol{x}$$

over all $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$. Solution: Precompute and store the values $\frac{1}{2}\|\boldsymbol{h}_1\|^2+\frac{1}{2}\|\boldsymbol{h}_2\|^2+\boldsymbol{h}_1^t\boldsymbol{h}_2$ in a matrix $\boldsymbol{M}=(m_{ij})$. For a given $\boldsymbol{x}$, compute all inner products $\boldsymbol{y}=\boldsymbol{H}^t\boldsymbol{x}$ involving $\boldsymbol{x}$. Then find the index pair (i,j) that minimize $m_{ij}-y_i-y_j$. This process is very fast since it reduces haplotyping to a search over the entries of a symmetric matrix. For optimal speed, the inner products $\boldsymbol{y}=\boldsymbol{H}^t\boldsymbol{x}$ are computed for all individuals simultaneously as a single matrix product $\boldsymbol{Y}=\boldsymbol{H}^t\boldsymbol{X}$. Also all inner products $\boldsymbol{h}_1^t\boldsymbol{h}_2$ can be recovered simultaneously from the matrix product $\boldsymbol{H}^t\boldsymbol{H}$. The squared norms $\|\boldsymbol{x}\|^2$ are irrelevant.

Fast Elimination of Redundant Haplotypes

The number of reference haplotypes may be in the tens of thousands. Within a genomic window, the number of unique haplotyes is small. How do we eliminate duplicates? If we suppose the number of SNPs within a window is a small power of 2, then the 0's and 1's of a haplotype can be interpreted as the binary digits of a nonnegative integer. Once these integers are sorted, it is trivial to eliminate redundancies. Conversion of the remaining unique integers back into binary vectors of 0's and 1's gives the haplotype matrix $\boldsymbol{H}$ for the window. Example of 4 SNPs and 3 reference haplotypes with haplotypes as columns:

$$\begin{pmatrix} 111 \\ 111 \\ 100 \\ 111 \end{pmatrix} \mapsto (15, 13, 13) \mapsto (13, 13, 15) \mapsto (13, 15) \mapsto \begin{pmatrix} 11 \\ 11 \\ 01 \\ 11 \end{pmatrix}$$

Using a permutation sort, no columns are actually moved.

Bridging Breaks

Occasionally, the two unique haplotypes assigned to two adjacent windows are inconsistent. In this situation there is a break and we must locate its position. Based on the sample (observed) SNPs alone, we slide the breakpoint across the two windows and determine its optimal position. For example, with two windows of length 8 each,

0000000000000000	extended left haplotype
1111111111111111	extended right haplotype
0000000001111111	extended observed haplotype
↑	optimal break position

The optimal break position minimizes the number of disagreements between the observed and reconstructed haplotypes. Note that left and right haplotypes are extended beyond their original windows by choosing consistent reference haplotypes.

Imputation of Untyped SNPs

The reference haplotypes may contain many more SNPs than the sample haplotypes, say 10^6 versus 10^7. Imputation of untyped SNPs is not routine because in a given window many reference haplotypes collapse to the same unique haplotype for the typed SNPs. Example:

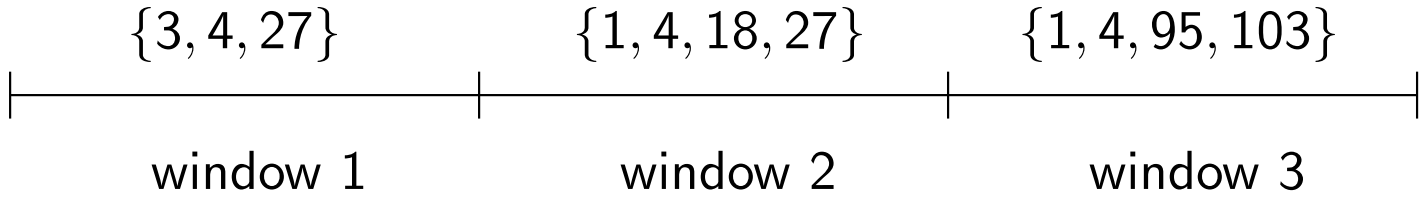


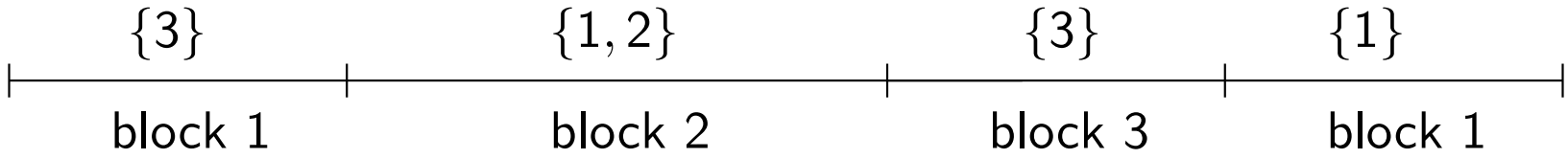
Figure: Consistent Reference Haplotype Sets in 3 Adjacent Windows

If no haplotype breaks occur across the 3 windows, then the only consistent reference haplotype is

$$\{4\} \quad = \quad \{3,4,27\} \cap \{1,4,18,27\} \cap \{1,4,95,103\}.$$

Resolving Ambiguities by Parsimony

What happens when the intersection principle fails to identify a unique reference haplotype? In such a situation we rely on parsimony and attempt to minimize the number of haplotypes employed per person per chromosome. Each reference haplotype is assigned a usage weight along a chromosome. In an unassigned block (region between breakpoints), the block is attributed to the consistent reference haplotype with largest weight. Example:



Weights of reference haplotypes:

$$w_1 = \frac{1}{2}2 + 1 = 2, \quad w_2 = \frac{1}{2}2 = 1, \quad w_3 = 1 + 1 = 2$$

Because $w_1 = 2 > w_2 = 1$, the ambiguous block 2 is assigned to reference haplotype 1.

Chromosome Painting

Local ancestry attribution can be achieved by haplotyping based on reference panels. Each haplotype block is assigned to a reference haplotype. If each reference haplotype is assigned to a country (or region or continent) of origin, then a person's genome can be painted block by block by country of origin. These labels can obviously be applied as predictors in association studies.

Data Compression

The sheer volume of SNP genotype data makes transferring data files a pain. Data compression can be achieved by haplotyping. Instead of sending genotypes, simply send each haplotype block start point and end point and a pointer to the relevant reference haplotype. These data immediately yield genotypes. A small list of errors of reconstruction complete the mailing package. This scheme depends on universal storage and curation of reference haplotypes. These should be stored on the cloud for easy access.

References

1. Chen GK, Wang K, Stram AH, Sobel EM, Lange K (2012) Mendel-GPU: Haplotyping and genotype imputation on graphics processing units. *Bioinformatics* 28:2979–2980
2. Chi EC, Chen GK, Zhou H, Ortega Del Vecchyo D, Lange K (2013) Genotype imputation via matrix completion. *Genome Res* 23:509–518
3. Hunter DR, Lange K (2004) A tutorial on MM algorithms. *American Statistician* 58:30–37
4. Lange K, Papp JC, Sinsheimer JS, Sobel EM (2014) Next generation statistical genetics: modeling, penalization, and optimization in high-dimensional data. *Annual Review Stat Applications* 1:279–300