Bayesian averaging for variational inference applied to genomic data - First Draft

William van Rooij - EPFL

7th June 2019

Contents

Chapter 1

Introduction

1.1 Situation

For the past years, data science has been increasingly present in the world. From financial establishments to road management companies, a lot of industry sectors are integrating data science in the way business is done. With the expansion of computer performance, we are able to implement faster computation and can work with more complex models. The volume of available data, hence analysable data, is also growing, which allows more accurate inference.

Often, when trying to find a model for data, we have many more observations than parameters to fit, a *large n, small p* situation. This is the most common type of statistical analysis. Bayesian hierarchical modelling is a strong tool to identify the dependencies across multiple sources of informations, but, the number of parameters may be much larger than the number of observations. This is often the case in genomic research, where the situation is called *small n, large p*. Traditional techniques do not then apply, because of both statistical and computational constraints.

In this report, we will focus on the *small n - large p* situation in the context of genetic association. We will focus on high dimensional Bayesian inference, with its statistical advantages and its computational problem that often dissuades users to adopt this solution in statistical applications.

1.2 Motivation

Current technology allows us to numerically represent the human genome, a whole new set of data is available to study the association between the genome and various diseases or phenotypes. Some of these newly available data are *genetic variants*, a change at a specific location on the genome (locus), where the different versions are called *alleles*. We will focus on the most common genetic variant, the *single nucleotide polymorphism* (SNP), a variation in the nucleotide that is present to some appreciable extent in the population. Some combinations of SNPs are inherited together, which yields block-wise dependence structures. We will calculate the association between SNPs and transcript expressions, called *traits*.

The strong correlation structure, observable in Figure 1.1, means that two SNPs in a same correlation block will be hard to differentiate. The goal is two represent the probabilities of association between a SNP and a trait, we should be able to observe the block-wise correlation in our results, the probabilities of association should be high even when the mode is not the real one.

We focus on *expression quantitative trait locus* (eQTL) analyses, which study the effects of genetic variants, in our case SNPs, on the expression of transcripts, or genes. The data used for eQTL studies consist generally of several hundred thousands SNPs and thousands of transcripts of expression outcomes. It is, in fact, a *small n, large p, large q* situation, where p is the number of SNPs and q is the number of transcripts of expressions.

Bayesian inference involves many integrals, but these usually need to be approximated. Markov Chain Monte Carlo (MCMC) algorithms are a standard technique for the approximation of integrals and can be fast and accurate when working on reasonably small datasets. When the dataset dimensions grow, however, MCMC algorithms become very time-consuming.

When performing MCMC inference, likelihoods and sometimes gradients need to be calculated at each iteration. The cost of these calculations increases with the number of parameters. Moreover, the more dimensions the problem has, the less accurate the approximations become, requiring more iterations to keep the precision needed. For the algorithm to end, all the parameters need to have converged, which means all parameters need to be checked and stored, which is often impossible when their number is very high.

In our situation, *small n, large p, large q*, the computational cost of using an MCMC algorithm is huge. The time and memory needed to run

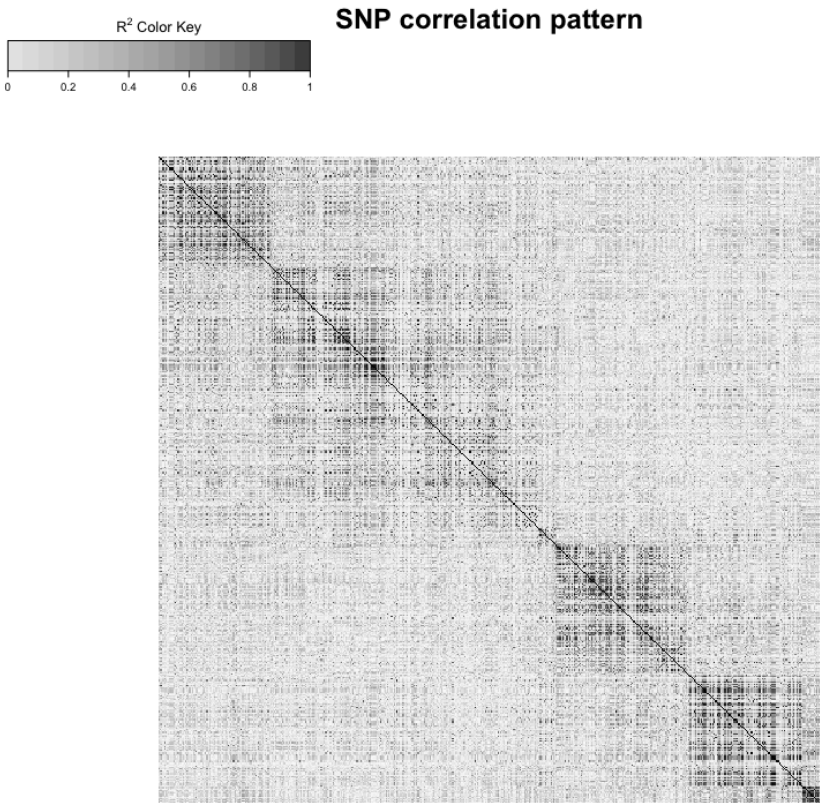

Figure 1.1: Visualisation of the correlation structure in the SNPs. We can clearly see the block-wise correlation structure.

the algorithm are not acceptable. We have to use an alternative solution, which we choose to be variational inference [2].

Chapter 2

Model

Our goal is to estimate the association between a SNP s and a trait t. To do so, we let $\boldsymbol{X} = (X_1, \ldots, X_p)$ be the design matrix, representing the SNPs, and $\boldsymbol{y} = (y_1, \ldots, y_q)$ be the response variables, representing the traits. The SNPs have a strong local correlation on the genome. We consider $\boldsymbol{y}$ as the response matrix and $\boldsymbol{X}$ as the candidate predictors of the linear model, where each response y_t is linearly related with the predictors $\boldsymbol{X}$ and has a residual precision $\boldsymbol{\tau}_t$, i.e.,

$$\boldsymbol{y}_{n\times q} = \boldsymbol{X}_{n\times p}\, \boldsymbol{\beta}_{p\times q} + \boldsymbol{\epsilon}_{n\times q}, \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \tau_t^{-1} I_n),$$

where $\boldsymbol{\beta}_{st}$ represents the association between SNP s and trait t. The parameters τ_t and σ^{-2} have the following prior distributions,

$$\begin{aligned} \tau_t &\sim \text{Gamma}(\eta_t, \kappa_t), \\ \sigma^{-2} &\sim \text{Gamma}(\lambda, \nu). \end{aligned}$$

We introduce $\boldsymbol{\gamma}_{p\times q}$, a binary matrix which says which pairs of SNPs and traits are associated. The SNP s and trait t are associated if and only if $\gamma_{st} = 1$. To enforce sparsity in $\boldsymbol{\beta}$, we set a "spike-and-slab" prior distribution on β_{st}, i.e.,

$$\beta_{st} \mid \gamma_{st}, \sigma^2, \tau_t \sim \gamma_{st}\, \mathcal{N}(0, \sigma^2 \tau_t^{-1}) + (1 - \gamma_{st})\, \delta_0,$$

where δ_0 is a Dirac distribution.

We call ω_s the parameter controlling to the proportion of responses associated with the predictor X_s. Then, the prior distribution of γ_{st} given ω_s is:

$$\gamma_{st} \mid \omega_s \sim \text{Bernoulli}(\omega_s).$$

We choose ω_s to follow a Beta distribution,

$$\omega_s \sim \text{Beta}(a_s, b_s),$$

with parameters a_s and b_s chosen to enforce sparsity. If we define $p^* \ll p$ as the expected number of predictors involved in the model, we want to set a_s and b_s such that the prior probability that X_s is associated with at least one response is equal to p^*/p. We fix the mean of the distribution but let the variance be free, the solution still has one degree of freedom so multiple solutions are possible, e.g.,

$$a_s = 1,\ b_s = q(p - p^*)/p^*.$$

We are interested in estimating the association between the SNPs and the traits, i.e. $\boldsymbol{\beta}$. It is common to estimate the parameters based on the observations $\boldsymbol{y}$, the associated density function is

$$\begin{aligned} p(\boldsymbol{\beta} \mid \boldsymbol{y}) &= \int \cdots \int p(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\omega}, \boldsymbol{\tau}, \sigma^{-2} \mid \boldsymbol{y}) \, \mathrm{d}\boldsymbol{\gamma} \, \mathrm{d}\boldsymbol{\omega} \, \mathrm{d}\boldsymbol{\tau} \, \mathrm{d}\sigma^{-2}, \\ &= \frac{1}{p(\boldsymbol{y})} \int \cdots \int p(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\omega}, \boldsymbol{\tau}, \sigma^{-2}) \, \mathrm{d}\boldsymbol{\gamma} \, \mathrm{d}\boldsymbol{\omega} \, \mathrm{d}\boldsymbol{\tau} \, \mathrm{d}\sigma^{-2}, \end{aligned}$$

with

$$\begin{aligned} p(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\omega}, \boldsymbol{\tau}, \sigma^{-2}) = &\left\{ \prod_{t=1}^{q} p(\boldsymbol{y}_t \mid \boldsymbol{\beta}_t, \tau_t) \right\} \left\{ \prod_{t=1}^{q} \prod_{s=1}^{p} p(\beta_{st} \mid \gamma_{st}, \tau_t, \sigma^{-2}) \right\} \left\{ \prod_{t=1}^{q} \prod_{s=1}^{p} p(\gamma_{st} \mid \omega_s) \right\} \\ &\times \left\{ \prod_{s=1}^{p} p(\omega_s) \right\} \left\{ \prod_{t=1}^{q} p(\tau_t) \right\} p(\sigma^{-2}), \end{aligned}$$

where, as mentioned earlier,

$$\begin{aligned} \boldsymbol{y}_t \mid \boldsymbol{\beta}_t, \tau_t &\sim \mathcal{N}_n \left(\boldsymbol{X}\boldsymbol{\beta}_t, \tau_t^{-1} \boldsymbol{I}_n \right), \\ \beta_{st} \mid \gamma_{st}, \tau_t, \sigma^{-2} &\sim \gamma_{st} \mathcal{N} \left(0, \sigma^2 \tau_t^{-1} \right) + (1 - \gamma_{st}) \delta_0, \\ \gamma_{st} \mid \omega_s &\sim \text{Bernoulli}(\omega_s), \\ \omega_s &\sim \text{Beta}(a_s, b_s), \\ \tau_t &\sim \text{Gamma}(\eta_t, \kappa_t), \\ \sigma^{-2} &\sim \text{Gamma}(\lambda, \nu), \end{aligned}$$

and δ_0 is the Dirac distribution.

Chapter 3

Variational Inference

When computing the posterior density of parameters $\boldsymbol{\theta}$ according to observed data $\boldsymbol{y}$, variational inference simplifies the computation by approximating the posterior density $p(\boldsymbol{\theta} \mid \boldsymbol{y})$ with a simpler density $q(\boldsymbol{\theta})$. It gives an approximation to the posterior distribution as a result of an optimization problem that minimizes a measure of "closeness" as objective function.

If we have observations $\boldsymbol{y}$ and parameters $\boldsymbol{\theta}$, we need to determine the posterior distribution of the parameters conditional on the observations $p(\boldsymbol{\theta} \mid \boldsymbol{y})$. Given a family of densities $\mathcal{D}$ over the parameters, we want to find the distribution $q \in \mathcal{D}$ that minimizes the "closeness" measure compared to $p(\boldsymbol{\theta} \mid \boldsymbol{y})$.

Variational inference minimizes the Kullback–Leibler divergence as a "closeness" measure. Introduced in 1951 by Kullback and Leibler[4], this is the most common divergence measure used in statistics and machine learning:

$$\mathrm{KL}(q \parallel p) := \int q(\boldsymbol{\theta}) \log \left(\frac{q(\boldsymbol{\theta})}{p(\boldsymbol{\theta} \mid \boldsymbol{y})} \right) \mathrm{d}\boldsymbol{\theta}.$$

It is described as a "directed divergence" as it is asymmetric, i.e., $\mathrm{KL}(p \parallel q) \neq \mathrm{KL}(q \parallel p)$.

Determining the family $\mathcal{D}$ can be difficult, as we need the family to be simple enough to be optimized efficiently, but flexible enough for the approximation $q \in \mathcal{D}$ to be close to $p(\boldsymbol{\theta} \mid \boldsymbol{y})$ with respect to the Kullback–Leibler divergence. The approximation will then be

$$q^*(\boldsymbol{\theta}) = \underset{q(\boldsymbol{\theta}) \in \mathcal{D}}{\arg\min}\, \mathrm{KL}\left[q(\boldsymbol{\theta}) \parallel p(\boldsymbol{\theta} \mid \boldsymbol{y}) \right].$$

Minimizing the Kullback–Leibler divergence can be complicated depending on the density p that we want to approximate and the density family $\mathcal{D}$ that we want q to be part of. We can decompose the Kullback–Leibler divergence as

$$\begin{aligned}\mathrm{KL}\left[q(\boldsymbol{\theta})||p(\boldsymbol{\theta} \mid \boldsymbol{y})\right] &= \mathbb{E}\left[\log q(\boldsymbol{\theta})\right] - \mathbb{E}\left[\log p(\boldsymbol{\theta} \mid \boldsymbol{y})\right] \\ &= \mathbb{E}\left[\log q(\boldsymbol{\theta})\right] - \mathbb{E}\left[\log p(\boldsymbol{y}, \boldsymbol{\theta})\right] + \log p(\boldsymbol{y}).\end{aligned}$$

We introduce the evidence lower bound:

$$\mathcal{L}(q) = \mathbb{E}\left[\log p(\boldsymbol{\theta}, \boldsymbol{y})\right] - \mathbb{E}\left[\log q(\boldsymbol{\theta})\right] = \int q(\boldsymbol{\theta}) \log \frac{p(\boldsymbol{y}, \boldsymbol{\theta})}{q(\boldsymbol{\theta})} \mathrm{d}\boldsymbol{\theta}.$$

When decomposing the Kullback–Leibler divergence, we obtain

$$\mathrm{KL}(q \parallel p) = \log(p) - \mathcal{L}(q).$$

This means that the Kullback–Leibler divergence is the difference between the marginal log-likelihood with no effect on the optimization and a function $\mathcal{L}(q)$. Hence, minimizing the Kullback–Leibler divergence is the same as maximizing $\mathcal{L}(q)$. The difference lies in the complexity of the problems, minimizing the Kullback–Leibler divergence is not tractable, but maximizing $\mathcal{L}(q)$ admits a closed form when the family of densities $\mathcal{D}$ is well chosen. In such a case, we prefer to use $\mathcal{L}(q)$ as an objective function.

Jensen's inequality provides another way to see that $\mathcal{L}(q)$ is a lower bound for the marginal log-likelihood, which is why we call it the evidence lower bound, or variational lower bound,

$$\begin{aligned}\log p(\boldsymbol{y}) &= \log \int p(\boldsymbol{y}, \boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta}, \\ &= \log \int \frac{p(\boldsymbol{y}, \boldsymbol{\theta})}{q(\boldsymbol{\theta})} q(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta}, \\ &\geq \int q(\boldsymbol{\theta}) \log \left\{ \frac{p(\boldsymbol{y}, \boldsymbol{\theta})}{q(\boldsymbol{\theta})} \right\} \mathrm{d}\boldsymbol{\theta}, \\ &= \mathcal{L}(q).\end{aligned}$$

Hence, $\log p(\boldsymbol{y}) \geq \mathcal{L}(q)$.

3.1 Mean-field approximation

The complexity of the optimization problem is directly bound to the complexity of the family of densities $\mathcal{D}$ to which $q(\boldsymbol{\theta})$ belongs. We introduce the

mean-field variational family, where the parameters are mutually independent.

Let $\{\theta_j\}_{j=1}^{J}$ be a partition of $\boldsymbol{\theta}$, $q \in \mathcal{D}$ and $\mathcal{D}$ a mean-field variational family, then,

$$q(\boldsymbol{\theta}) = \prod_{j=1}^{J} q_j(\theta_j).$$

We determine the variational factors $q_j(\theta_j)$ by maximizing $\mathcal{L}(q_j)$. Hence, the variational family does not directly represent the observed data, they are both linked through the optimization of the evidence lower bound.

Concretely, we assume the independence of most of the parameters,

$$q(\boldsymbol{\theta}) = \left\{ \prod_{s=1}^{p} \prod_{t=1}^{q} q(\beta_{st}, \gamma_{st}) \right\} \left\{ \prod_{s=1}^{p} q(\omega_s) \right\} \left\{ \prod_{t=1}^{q} q(\tau_t) \right\} q(\sigma^{-2}).$$

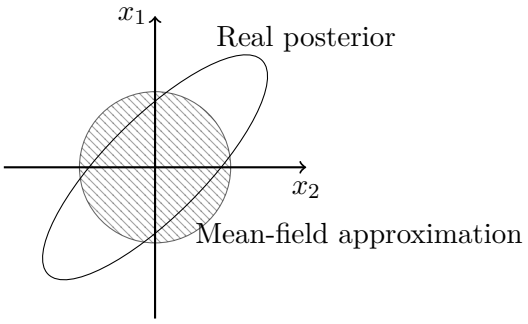

Figure 3.1: To visualise the mean-field approximation, we consider a two dimensional Gaussian distribution, represented in clear in Figure 3.1. The mean-field approximation of the posterior distribution is represented by the barred circle. We see that the mean of the approximation is the same as the real mean, but the covariance does not match the covariance of the real posterior.

We have transformed, using the evidence lower bound and the mean-field approximation our problem into a optimization problem. We now need a way to solve this problem. In the following section, we describe the coordinate ascent algorithm.

3.2 Coordinate ascent algorithm

Coordinate ascent mean-field variational inference is one of commonly used to solve this optimization problem. The algorithm iterates on the parameters

of the mean-field approximation, optimizing them one at the time. It yields a local optimum for the evidence lower bound. The algorithm is based on the following result:

Lemma 3.1 *If we fix $q_l(\theta_l)$, $l \neq j$, then the optimal $q_j^*(\theta_j)$ satisfies:*

$$q_j^*(\theta_j) \propto \exp\left\{\mathbb{E}_{-j}\left[\log p(\theta_j \mid \boldsymbol{\theta}_{-j}, \boldsymbol{y})\right]\right\}.$$

Where $\mathbb{E}_{-j}$ denotes the expectation with respect to all θ_l, $l \neq j$.

Based on this result, the algorithm updates one parameter θ_j at a time while the others stay fixed. The algorithm stops when $\mathcal{L}(q)$ increases by less than a pre-determined threshold ε.

Algorithm 1: Coordinate ascent variational inference

input : $p(\boldsymbol{y}, \boldsymbol{\theta})$, dataset y tolerance ε

output : $q(\boldsymbol{\theta}) = \prod_{j=1}^{J} q_j(\theta_j)$

initialize: $q_j(\theta_j)$

repeat

  for $j \in \{1, \ldots, J\}$ **do**

    set $q_j(\theta_j) \propto \exp\left\{\mathbb{E}_{-j}\left[\log p(\theta_j \mid \boldsymbol{\theta}_{-j}, \boldsymbol{y})\right]\right\}$

  $\mathcal{L}^{\mathsf{old}}(q) \leftarrow \mathcal{L}(q)$

  $\mathcal{L}(q) \leftarrow \mathbb{E}\left[\log p(\boldsymbol{\theta}, \boldsymbol{y})\right] - \mathbb{E}\left[\log q(\boldsymbol{\theta})\right]$

until $|\mathcal{L}^{\mathsf{old}}(q) - \mathcal{L}(q)| < \varepsilon$

return $q(\boldsymbol{\theta})$

At every iteration, $\mathcal{L}(q)$ is guaranteed to increase. The algorithm yields a local optimum depending on the initialization of the $q_j(\theta_j)$, $j = 1, \ldots, J$. Having different initializations could yield different optima that correspond to different models.

In our case, the posterior distributions of our model's parameters are:

$$\begin{aligned}
\beta_{st} \mid \gamma_{st} = 1, \boldsymbol{y} &\sim \mathcal{N}\left(\mu_{\beta,st}, \sigma^2_{\beta,st}\right), \\
\beta_{st} \mid \gamma_{st} = 0, \boldsymbol{y} &\sim \delta_0, \\
\gamma_{st} \mid \boldsymbol{y} &\sim \text{Bernoulli}(\gamma_{st}^{(1)}), \\
\omega_s \mid \boldsymbol{y} &\sim \text{Beta}(a_s^*, b_s^*), \\
\tau_t \mid \boldsymbol{y} &\sim \text{Gamma}(\eta_t^*, \kappa_t^*), \\
\sigma^{-2} \mid \boldsymbol{y} &\sim \text{Gamma}(\lambda^*, \nu^*).
\end{aligned}$$

Chapter 4

Multimodality

Bayesian model averaging is a strategy to account for multiple competing models in an inference problem. It consists of weighting the different models in a weighted average with the probability that the data corresponds to each model. The more the model corresponds to the observed data, the more it will stand out in the result.

Assume that the data $\boldsymbol{y}$ correspond to multiple models M_k, $k = 1, \ldots, K$, and Δ is the quantity of interest. We have the posterior distribution:

$$p(\Delta \mid \boldsymbol{y}) = \sum_{k=1}^{K} p(\Delta \mid M_k, \boldsymbol{y})\, p(M_k \mid \boldsymbol{y}). \tag{4.1}$$

This corresponds to a weighted average of the posterior distribution under each of the considered models with weights corresponding to the posterior models probabilities.

Instead of $p(\Delta \mid \boldsymbol{y})$ in Equation 4.1, we might be interested in approximating:

$$\mathbb{E}\left[\Delta \mid \boldsymbol{y}\right] = \sum_{k=1}^{K} \mathbb{E}\left[\Delta \mid M_k, \boldsymbol{y}\right]\ p(M_k \mid \boldsymbol{y}).$$

The posterior probability for model M_k is given by:

$$p(M_k \mid \boldsymbol{y}) = \frac{p(\boldsymbol{y} \mid M_k)\, p(M_k)}{\sum_{j=1}^{K} p(\boldsymbol{y} \mid M_j)\, p(M_j)}, \tag{4.2}$$

where $p(\boldsymbol{y} \mid M_k)$ is the likelihood of model M_k, and $p(M_k)$ is the prior probabilities of the model M_k. It can, for example, depend on the complexity

of the model, to favour the simpler models, or, if we consider the models to be equiprobable, it would be equal to $p(M_k) = 1/K$, $k = 1, \ldots, K$.

We know that the evidence lower bound and the Kullback–Leibler divergence are related and that minimizing the Kullback–Leibler divergence is equivalent to maximizing the evidence lower bound, and that they verify:

$$\mathrm{KL}(q \parallel p) = \log p(\boldsymbol{y}) - \mathcal{L}(q).$$

Since we minimized the Kullback–Leibler divergence, we can use $\mathcal{L}(q)$ as an approximation for $\log p(\boldsymbol{y})$ in Equation 4.2.

Our quantity of interest is γ_{st}, i.e. we want to know if the SNP s and the trait t are associated. Using Algorithm **??**, we initialise the distributions $q_j(\theta_j)$ with different starting points, and consider the optimums yielded by the algorithm.

We can consider each optimum to be a model representing the data, and we can apply a form of Bayesian model averaging to combine them all using the method we described here above. We approximate $\log p(\boldsymbol{y})$ by $\mathcal{L}(q)$ in Equation 4.2, and obtain an approximation for $\mathbb{E}[\gamma_{st} \mid \boldsymbol{y}]$ considering all the models we have obtained in the algorithm.

As we are dealing with strongly correlated structures, some modes will be strongly plausible even if they are not the real mode. This incertitude should be visible when observing the resulting approximations for $\mathbb{E}[\gamma_s t \mid \boldsymbol{y}]$. Indeed, we should see the real mode standing out, but we should have some other modes visible as well, more or less visible according to their plausibility.

4.1 Simulated Annealing

To identify data dependence structures, instead of altering the model, we can change the inference strategy. When dealing with highly correlated data, our coordinate ascent algorithm often gets stuck in local modes. We use a simulated annealing procedure to augment our method and improve the modes exploration.

We start with the same strategy as earlier, i.e. minimizing the reverse Kullback–Leibler divergence,

$$\mathrm{KL}(q \parallel q) = -\int q(\boldsymbol{\theta}) \log \left\{ \frac{p(\boldsymbol{\theta} \mid \boldsymbol{y})}{q(\boldsymbol{\theta})} \right\} \mathrm{d}\boldsymbol{\theta}.$$

we end up with the lower bound as objective function,

$$\mathcal{L}(q) = \mathbb{E}_q \left[\log p(\boldsymbol{y}, \boldsymbol{\theta}) \right] - \mathbb{E}_q \left[\log q(\boldsymbol{\theta}) \right],$$

which is composed of the expected log joint distribution, which motivates the approximation to take more mass where the variables explain more the data, and the entropy, that encourages the dispersion of the approximation.

The idea of simulated annealing is to introduce a temperature T that yields a series of heated distributions,

$$p_T(\boldsymbol{y}, \boldsymbol{\theta}) \propto p(\boldsymbol{y}, \boldsymbol{\theta})^{1/T},$$

and influences the differences of the modes. The temperature starts high, smoothing the density of interest, and gets lower along the process until it reaches the original density. The high temperatures yield an easier search for the global optimum. The temperature multiplies the entropy term, allowing for more disparate approximations to be considered,

$$\mathcal{L}_T(q) = \int q_T(\boldsymbol{\theta}) \log p(\boldsymbol{y}, \boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta} - T \int q_T(\boldsymbol{\theta}) \log q_T(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta}, \; T \geq 1, \qquad (4.3)$$

where q_T is the heated variational distribution, it applies a penalty on the log joint distribution when the temperature $T > 1$, and relaxes the penalty as T goes down until $T = 1$, where the penalty becomes null.

With the same process we used without the annealing, we can write (4.3) with respect to θ_j as

$$\mathcal{L}_T(q) = \mathbb{E}_j \left[\mathbb{E}_{-j} \left\{ \log p(\boldsymbol{y}, \boldsymbol{\theta}) \right\} - T \log q_T(\theta_j) \right] + \text{const},$$

that can be written as

$$\mathcal{L}_T(q) = T \mathbb{E}_j \left[\log \left\{ \frac{p_{T,-j}(\boldsymbol{y}, \theta_j)}{q_T(\theta_j)} \right\} \right] + \text{const},$$

where $p_{T,-j}(\boldsymbol{y}, \theta_j) \propto \exp\left\{T^{-1}\mathbb{E}_{-j}\left[\log p(\boldsymbol{y}, \boldsymbol{\theta})\right]\right\}$, $\mathbb{E}_j$ is the expected value with respect to $q_T(\theta_j)$, $\mathbb{E}_{-j}$ is the expected value with respect to every $q_T(\theta_k)$ where $k \neq j$, and const is independent of v_j.

$\mathcal{L}_T(q)$ is maximal when $q_T(\theta_j) = p_{T,-j}(\boldsymbol{y}, \theta_j)$, which is equivalent to when

$$\log q_T(\theta_j) = T^{-1}\mathbb{E}_{-j}\left[\log p(\boldsymbol{y}, \boldsymbol{\theta})\right] + \text{const}\,, \quad j = 1, \ldots, J.$$

We have different options for the temperature schedule including a geometric spacing,

$$T_l = (1 + \Delta)^{l-1}, \quad \Delta = T_L^{1/(L-1)} - 1,$$

an harmonic spacing,

$$T_l = 1 + \Delta(l - 1), \quad \Delta\frac{T_L - 1}{L - 1},$$

and a linear spacing,

$$T_l^{-1} = T_L^{-1} + \Delta(L - l), \quad \Delta = \frac{1 - T_L^{-1}}{L - 1},$$

where $l = 1, \ldots, L$ and T_L is the hottest temperature. T_l is the temperature used at step l and L is the number of steps necessary to lower the temperature to the initial temperature $T = 1$, where the initial algorithm is ran until convergence.

To cope with strongly correlated structures and represent the incertitude of the modes, we use simulated annealing combined with our weighted average and retrieve a combination of different models yielded from different initialisations. However, two different initialisations that gave different modes could give the same mode when using simulated annealing as the density function is "smoothed" by the temperature. The number of different modes considered in the weighted average will hence be less than when not performing the simulated annealing step before hand.

Chapter 5

Simulations

In her R-package `locus`, H. Ruffieux has implemented a function `locus` that estimates the probabilities of association between a SNP and a trait. We will use this function to build our own method and also to have a comparison. If our method would not perform better than this implementation, it because irrelevant.

Our method is basically calling multiple times that `locus` function and combine all the results in an weighted average. For each call, we initialized the parameters differently, and hoped to obtain different optimums. We will call this method "multiple locus", the single call method will be "single locus" and when augmented with simulated annealing, we will just add "annealing" before the name.

We have drawn at random the initial parameters for the optimal approximations $q^*(\boldsymbol{\theta})$. We have used H. Ruffieux's function `locus` to calculate the probabilities of association between the SNPs and the traits, as well as the evidence lower bound for each initialisation. Then we used the evidence lower bounds as weights in our variant of Bayesian model averaging to combine the results of each initialisation.

We first tested our method on generated data, to be able to compare the results calculated with the truth. We have used H. Ruffieux's R-package `echoseq` to generate block wise strongly autocorrelated SNPs and traits, as well as their associations. We have generated 300 observations of 500 SNPs, with autocorrelations between 0.95 and 0.99, by blocks of 10 SNPs. It is important to note that the correlations defined here are correlations between the underlying variables, so the correlations between the SNPs are a little bit weaker. As we want to visualise the probabilities of association, we generated just one trait. We have selected five SNPs to be associated

with the trait, for better visualisation, all five SNPs are in the 50 first SNPs.

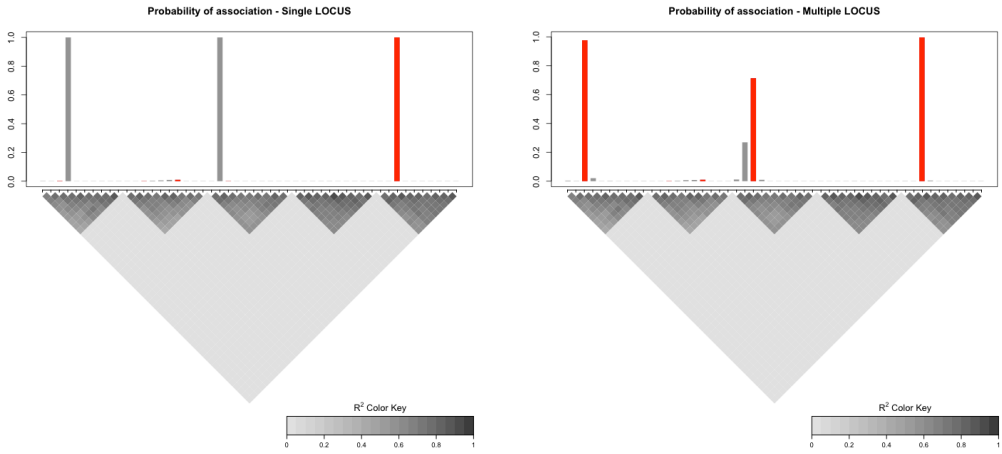

Figure 5.1: Probabilities of association of the 50 first SNPs with a single trait calculated with a single call of the `locus` function (left) and when doing a weighted average on multiple calls of the `locus` function (right). In red are the five real associated SNPs. Underneath are the correlations between the different SNPs, they are the same for the two sides as the SNPs used are the same.

In Figure 5.1, we have plotted the probabilities of association of the 50 first SNPs, out of 500 used, with a single trait t. In the construction of the data, we have enforced the five real associated SNPs to be in the 50 first SNPs, for observations reasons, they are marked in red. The real associated SNPs are the same for both of the plots.

On the left, we have used a single call of the `locus` function, it is equivalent to choosing a single model M and calculating

$$\mathbb{E}\left[\gamma_{st} \mid \boldsymbol{y}\right] = \mathbb{E}\left[\gamma_{st} \mid M, \boldsymbol{y}\right] \; p\left(M \mid \boldsymbol{y}\right).$$

On the right, we have used the weighted averaging method over a range of 100 different initial parameters yielding K different models M_k, $k = 1 \ldots, K$. We then calculated

$$\mathbb{E}\left[\gamma_{st} \mid \boldsymbol{y}\right] = \sum_{k=1}^{K} \mathbb{E}\left[\gamma_{st} \mid M_k\right] \; p\left(M_k \mid \boldsymbol{y}\right).$$

We can see that when using a single call of the `locus` function, the SNPs found to be associated with the trait are not the right ones. This can be explained by the strong correlations in the block structure. The strong correlations can mislead the function into yielding the wrong SNP in the same correlation block.

When using a weighted average on multiple models yielded by multiple initialisations the function `locus`, we can see that we consider different models containing the one we used to build the data. We can see that three of the five real SNPs associated with the trait are found by the algorithm.

We can also see that the two grey pikes of the left plot can be seen on the right plot. This can be explained by the fact that the model found on the left plot has been considered in the weighted average of the right plot. Apparently, a few other iterations of the function have also mislead the SNP corresponding to the second spike of the left plot to be associated with the trait, as its probability of association is non negligible. The first spike of the left plot is not as present in other occurrences of the function as we can see that it is considerably small.

The block wise correlation structure is also visible in the probabilities of association for the weighted average method. We can see that four SNPs of the middle block have all non null probabilities of association with the trait. This phenomena can be explained by the strong correlations between these SNPs misleading the algorithm into designating a wrong SNP with a strong correlation to the right one.

We wanted to compare four methods, single locus, multiple locus and their simulated annealing augmented counterparts. We chose four different situations that we thought would be interesting to compare. Two of the settings were constituted of 15 associated SNPs, whereas the two other had 50 associated SNPs. We also had a pair of settings where the proportion of the response variance that could be explained by the SNPs could be up to 50% and, for another pair, up to 80%.

The simulated annealing augmented methods have an initial temperature fixed at $T_L = 2$, we have chosen a geometric spacing with ten steps.

In Figure 5.2 are represented the ROC curves of the four methods we wanted to compare, for each of the four settings we mentioned earlier. We have truncated the ROC curves as we are interested only in the accuracy of the methods for a small error rate. We have the same settings a we had for Figure 5.1, as well as just one trait. To fully check the accuracy of the different methods, we could calculate the association of SNPs with more traits.

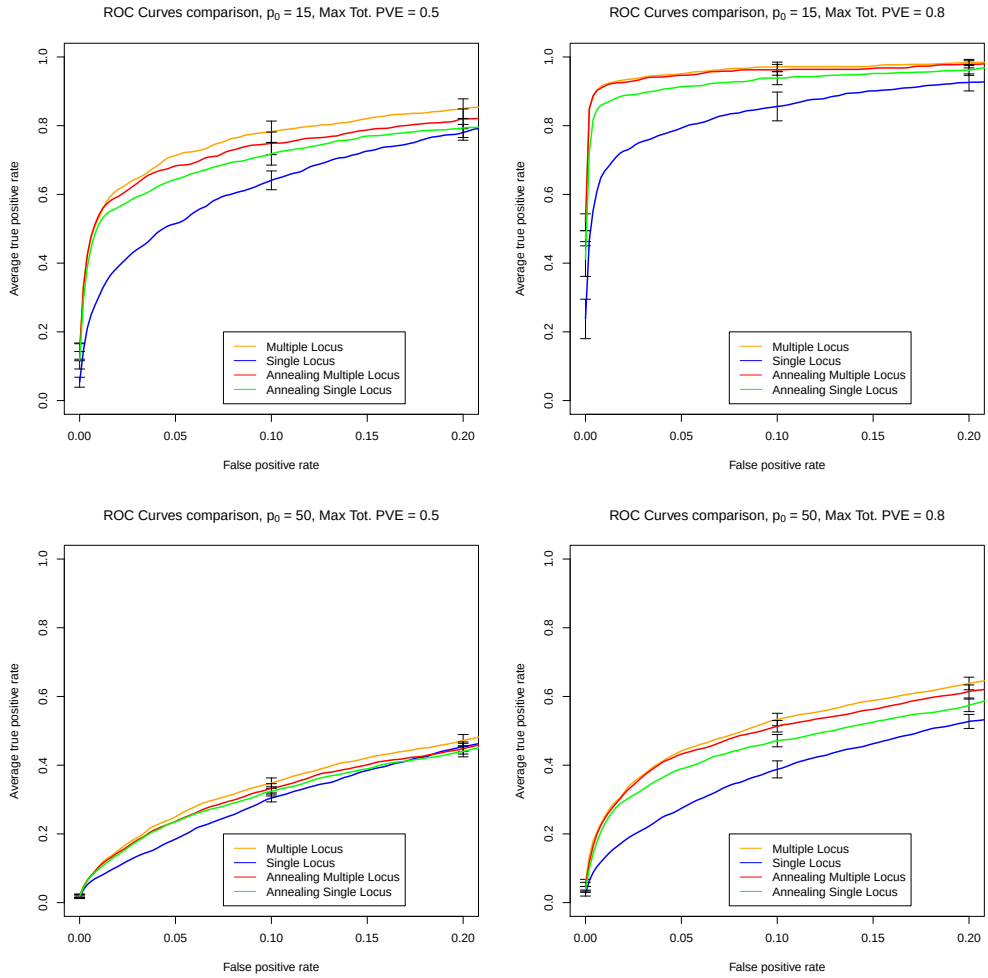

Figure 5.2: Comparison of ROC curves between multiple and single locus, and the same two methods augmented with a simulated annealing step, colored orange, blue, red, and green respectively. Top row: $p_0 = 15$, Left column: Max tot. PVE= 0.5, Bottom row: $p_0 = 50$, Right column: Max tot. PVE= 0.8

Firstly, we compare the single locus and the multiple locus methods. We can clearly see in Figure 5.2 that the multiple locus is more accurate than the single locus. The multiple locus considers many different modes, in our case 100, and attributes to each mode a weight associated to the likelihood of the data being obtained from said mode. We hope that the real mode is contained in the considered 100 modes, then the likelihood of the data originating from said mode will be high and the real associated SNPs will be more represented.

Secondly, we can see that when starting both the single locus and the multiple locus with a simulated annealing step, the multiple locus is still more accurate than the single locus, although the difference is smaller than it is without the simulated annealing step. This means that the annealing step does not prevent the multiple locus to consider multiple different models. It also means that the simulated annealing augmented single locus does not yield the right model every time. This could be because the initial temperature we have chosen was not high enough to smooth the densities enough to access the right modes.

Thirdly, the augmented single locus method is more accurate than the single locus method. The simulated annealing step allows the method to reach modes that cannot be reached by the single locus method with certain starting parameters. This allows the real mode to be reached more often when using a simulated annealing step, hence, to be more accurate.

Fourthly, we can see that the setting where we are looking for 50 SNPs associated with the trait and a maximum proportion of the response variance explained by the SNPs is 50% have all the methods similarly accurate. The accuracy "gained" when adding the simulated annealing step or when averaging over multiple initialisations is not relevant.

Finally, we see that the augmented multiple locus is very close in accuracy to multiple locus. The advantage of starting with a simulated annealing step is not necessarily yielding more accurate results. Based on the graphs, we can even say it might be less accurate. The simulated annealing step might diminish the number of modes considered for the average, putting more weight in the wrong models, hence leading the algorithm on the wrong mode.

It should be noted that for our method, should it be simulated annealing augmented or not, paralleled computation is possible, which can drastically diminish the time needed to compute it. Even if the method has to wait until the last iteration to converge, we would still be quicker than calculating the iterations one after the other.

Instead of comparing the accuracy of different methods, we now want to assess the accuracy of our method compared to simulated values. To do so, we have generated some data with H. Ruffieux's `echoseq` `R`-package, and extracted the matrix $\boldsymbol{\beta}$ to have the real parameters. We have simulated 300 observations of four SNPs, with a equicorrelated SNPs with a strong correlation of 0.955. A strong correlation is what can induce an error in the selection of the associated SNPs, and in verifying the accuracy of our methods, it is necessary to test in extreme situations.

We compare the posterior distributions of the parameters estimations obtained from our methods with posteriors distributions obtained from MCMC inference. The two inference methods have a different convergence and stooping criteria, so the comparison should be studied prudently. Our method is based on variational inference, which has a convergence criterion defined as a tolerance to be given. The MCMC inference does not necessarily visit the whole model space, so to counter that problem, we run 10^5 iterations and burn the first half.

To be able to perform MCMC on our data, we have chosen the number of parameters to be small, i.e. $p = 4$, $q = 1$. We are interested in evaluating the posterior distributions of $\boldsymbol{\beta} = (\beta_1, \beta_2, \beta_3, \beta_4)$. In the construction of our data, we have chosen $\beta_2, \beta_3 = 0$ and $\beta_1, \beta_4 \neq 0$.

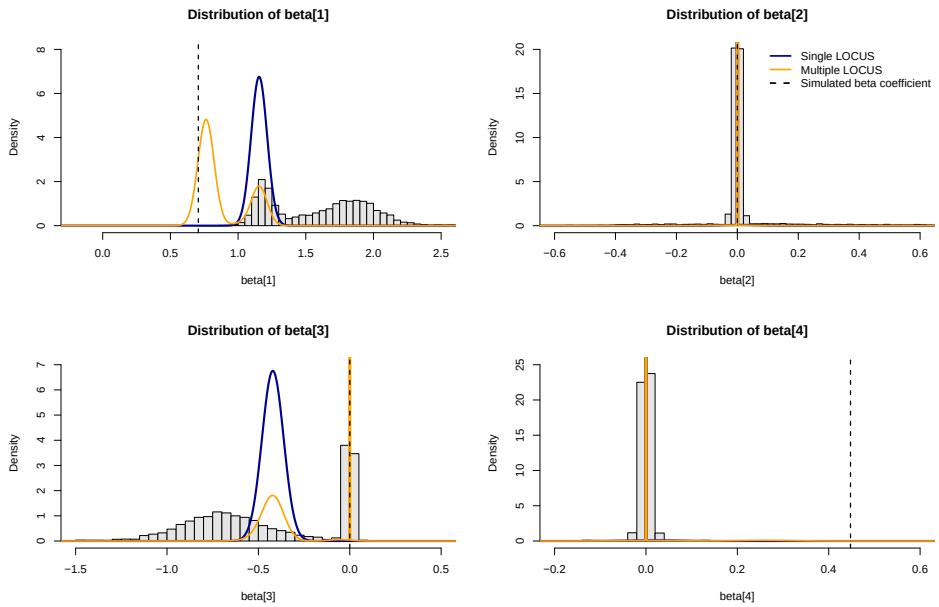

Figure 5.3: Comparison of single (blue) and multiple (orange) locus calculated posteriors for $\boldsymbol{\beta}$, MCMC simulated (histograms) $\boldsymbol{\beta}$ posteriors as well as the real (dashed black line) $\boldsymbol{\beta}$ values.

In Figure 5.3, we have plotted single and multiple locus calculated posteriors of $\boldsymbol{\beta}$, as well as the histogram of the MCMC simulated posteriors and the real values of $\boldsymbol{\beta}$. The orange and blue lines of β_2 and β_4 are superposed.

Firstly, we can see that multiple locus finds the right distribution every time but once, for β_4, where it finds the same estimation as the MCMC method, and the single locus method. However, the single locus only finds

the right estimation once, for β_2. This confirms what we read in the ROC curves of Figure 5.2, where we saw that the accuracy of the multiple locus is better that the single locus.

Secondly, when the single and multiple locus do not yield the same value for the parameters, the result of single locus is visible in the distribution of multiple locus. This is given by the fact that multiple locus considers the mode obtained from single locus in its averaging, and in that case, the mode obtained from single locus was relevant.

Chapter 6

Next steps

For the remainder of this project, we will compare the accuracy of our approach and its computational cost to other methods, such as annealing and non-weighted averaging for strong correlations. We will also try to combine annealing with our method.[**?**]

To be able to tell if our algorithm adequately explores the local modes, we will represent them with the level curves similarly as V. Rockova [3] did.

We plan to optimize the code that we implemented, to have an acceptable comparison with the other methods commonly used. If the results are satisfactory, we may include the function in H. Ruffieux's R-package (http://github.com/hruffieux/locus).

Finally, to be able to apply this method on real-life data would be the target of the whole project.

Bibliography

[1] B. Lewin, J. Krebs, S. T. Kilpatrick, and E. S. Goldstein. *Lewin's Genes*, volume 10. Jones and Bartlett, Sudbury, United States, 2011.

[2] David M. Blei, Alp Kucukelbir, Jon D. McAuliffe. Variational inference: A review for statisticians, 2018.

[3] Veronika Rocková. Particle em for variable selection, 2017.

[4] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22:79–86, 1951.