Title

Fast Empirical Bayesian LASSO for Multiple Quantitative Trait Locus Mapping

Permalink

https://escholarship.org/uc/item/5th101q9

Authors

Cai, Xiaodong
Huang, Anhui
Xu, Shizhong

Peer reviewed

RESEARCH ARTICLE

Fast empirical Bayesian LASSO for multiple quantitative trait locus mapping

Xiaodong Cai[1*], Anhui Huang[1] and Shizhong Xu[2]

Abstract

Background: The Bayesian shrinkage technique has been applied to multiple quantitative trait loci (QTLs) mapping to estimate the genetic effects of QTLs on quantitative traits from a very large set of possible effects including the main and epistatic effects of QTLs. Although the recently developed empirical Bayes (EB) method significantly reduced computation comparing with the fully Bayesian approach, its speed and accuracy are limited by the fact that numerical optimization is required to estimate the variance components in the QTL model.

Results: We developed a fast empirical Bayesian LASSO (EBLASSO) method for multiple QTL mapping. The fact that the EBLASSO can estimate the variance components in a closed form along with other algorithmic techniques render the EBLASSO method more efficient and accurate. Comparing with the EB method, our simulation study demonstrated that the EBLASSO method could substantially improve the computational speed and detect more QTL effects without increasing the false positive rate. Particularly, the EBLASSO algorithm running on a personal computer could easily handle a linear QTL model with more than 100,000 variables in our simulation study. Real data analysis also demonstrated that the EBLASSO method detected more reasonable effects than the EB method. Comparing with the LASSO, our simulation showed that the current version of the EBLASSO implemented in Matlab had similar speed as the LASSO implemented in Fortran, and that the EBLASSO detected the same number of true effects as the LASSO but a much smaller number of false positive effects.

Conclusions: The EBLASSO method can handle a large number of effects possibly including both the main and epistatic QTL effects, environmental effects and the effects of gene-environment interactions. It will be a very useful tool for multiple QTL mapping.

Background

Quantitative traits are usually controlled by multiple quantitative trait loci (QTLs) and environmental factors. Interactions among QTLs or between genes and environmental factors make a substantial contribution to variation in complex traits [1]. The goal of QTL mapping is to infer genomic loci that are associated with the trait and to estimate the genetic effects of these loci including their main effects and gene-gene (epistasis) and gene-environment ($G \times E$) interactions. Due to the physical linkage of and/or epistatic interactions among multiple QTLs, it is highly desirable to analyze a large number of loci simultaneously. Since hundreds or thousands of genomic loci or markers are usually genotyped

and involved in QTL mapping studies, including all these markers and their possible interactions in a model leads to a huge number of model variables, typically much larger than the sample size. Two general techniques often employed to handle such oversaturated models are variable selection and shrinkage.

Variable selection attempts to identify a subset of all possible genetic effects that best explain the phenotypic variation, typically using a stepwise search procedure in conjunction with a selection criterion such as the Bayesian information criterion (BIC) [2]. On the other hand, a shrinkage method includes all variables in the model but uses a penalty function of the variables or appropriate prior distributions on the variables to shrink most non-effect variables toward zero. Early shrinkage methods include ridge regression [3] and the least absolute shrinkage and selection operator (LASSO) [4]. More recently, Bayesian shrinkage method [5] has received

* Correspondence: x.cai@miami.edu
[1]Department of Electrical and Computer Engineering, University of Miami, Coral Gables, FL 33146, USA
Full list of author information is available at the end of the article

considerable attention and been applied to multiple QTL mapping [6-10]. All these works employ Markov chain Monte Carlo (MCMC) simulation to fit the Bayesian model and provide comprehensive information about the model drawing from the posterior distribution of the model variables. Despite the advances in the development of the MCMC simulation algorithms [11], MCMC simulations are computationally intensive and time consuming.

In order to reduce the computational burden of the fully Bayesian approach relying on MCMC simulation, one of the authors of this paper developed an empirical Bayes (EB) method [12] that uses a properly chosen prior distribution for the model variables to shrink variables toward zero. It was demonstrated that the EB method can handle a large number of model variables simultaneously. More recently, the EB method has been extended to handle classification predictor variables [13]. Although the EB method [12] requires much less computation comparing to the fully Bayesian approach, its efficiency is limited by the fact that a numerical optimization algorithm such as the simplex algorithm [14] is needed to estimate the variance components. On the other hand, a very efficient EB method, named relevance vector machine (RVM), for learning a linear model was developed in the machine learning community [15,16]. The RVM can estimate the variance components in a closed form, which along with other algorithmic techniques make it a very fast algorithm. The RVM assumes a uniform prior distribution for the variance components. Although this choice of the prior distribution gets rid of any hyperparameters to be pre-specified, it lacks the flexibility of adjusting the degree of shrinkage needed for analyzing a specific data set. Particularly, its uniform prior distribution may not provide enough shrinkage in multiple QTL mapping that includes a very large number of possible effects, often resulting in a large number of false effects [13].

In this paper, capitalizing on the idea of RVM, we developed a fast empirical Bayesian LASSO (EBLASSO) algorithm based on the Bayesian LASSO model [10,17] with an exponential prior distribution for the variance components in contrast to the inverse chi-square distribution for the variance components used by the EB method [12]. Simulation studies demonstrate that our EBLASSO method can provide a speed up to orders of magnitude faster than the EB method and can detect more true QTL effects without increasing the false positive rate. Real data analysis also show that the EBLASSO method is able to detect some effects when the EB method fails.

Methods
Linear model of multiple QTLs
Let y_i be the phenotypic value of a quantitative trait of the ith individual in a mapping population. Suppose we

observe y_i, $i = 1, ..., n$ of n individuals and collect them into a vector $\mathbf{y} = [y_1, y_2, ..., y_n]^T$. Considering environmental effects, main and epistatic effects of all markers and gene-environment ($G \times E$) interactions, we have the following linear regression model for $\mathbf{y}$:

$$\mathbf{y} = \mu + \mathbf{X}_E \boldsymbol{\beta}_E + \mathbf{X}_G \boldsymbol{\beta}_G + \mathbf{X}_{GG} \boldsymbol{\beta}_{GG} + \mathbf{X}_{GE} \boldsymbol{\beta}_{GE} + \mathbf{e}, \quad (1)$$

where μ is the population mean, vectors $\boldsymbol{\beta}_E$ and $\boldsymbol{\beta}_G$ represent the environmental effects and the main effects of all markers, respectively, vectors $\boldsymbol{\beta}_{GG}$ and $\boldsymbol{\beta}_{GE}$ capture the epistatic effects and the $G \times E$ interactions, respectively, $\mathbf{X}_E$, $\mathbf{X}_G$, $\mathbf{X}_{GG}$ and $\mathbf{X}_{GE}$ are the corresponding design matrices of different effects, and $\mathbf{e}$ is the residual error that follows a normal distribution with zero-mean and covariance $\sigma_0^2 \mathbf{I}$. Throughout the paper we use $\mathbf{I}$ to denote an identity matrix whose size can be clearly identified from the context.

The design matrix $\mathbf{X}_G$ depends on a specific genetic model. We adopt the widely used Cockerham genetic model as also used by [18] in their generalized linear model for multiple QTL mapping. For a back-cross design, the Cockerham model defines the values of the main effect of a marker as -0.5 and 0.5 for two genotypes at the marker. For an intercross (F_2) design, there are two possible main effects named additive and dominance effects. The Cockerham model defines the values of the additive effect as -1, 0 and 1 for the three genotypes and the values of the dominance effect as -0.5 and 0.5 for homozygotes and heterozygotes, respectively. The columns of the design matrix $\mathbf{X}_{GG}$ are obtained as the element-wise product of any two different columns of $\mathbf{X}_G$, and similarly the columns of $\mathbf{X}_{GE}$ are constructed as the element-wise product of any pair of columns from $\mathbf{X}_E$ and $\mathbf{X}_G$.

Defining $\boldsymbol{\beta} = [\boldsymbol{\beta}_E^T, \boldsymbol{\beta}_G^T, \boldsymbol{\beta}_{GG}^T, \boldsymbol{\beta}_{GE}^T]^T$, $\mathbf{X} = [\mathbf{X}_E, \mathbf{X}_G, \mathbf{X}_{GG}, \mathbf{X}_{GE}]$, we can write (1) in a more compact form:

$$\mathbf{y} = \mu + \mathbf{X}\boldsymbol{\beta} + \mathbf{e}. \quad (2)$$

Suppose that there are p environmental covariates and q markers whose main effects are additive, then the size of matrix $\mathbf{X}$ is $n \times k$ where $k = p + q(q + 1)/2 + pq$. Typically, we have $k \gg n$. If dominance effects of the markers are considered, k is even larger. Our goal is to estimate all possible environmental and genetical effects on $\mathbf{y}$ manifested in the regression coefficients $\boldsymbol{\beta}$, which is a challenging problem because $k \gg n$. However, we would expect that most elements of $\boldsymbol{\beta}$ are zeros and thus we have a sparse linear model. Taking into account this sparsity, we will adopt the Bayesian LASSO model [10] where appropriate prior distributions are assigned to the elements of $\boldsymbol{\beta}$ as described in the next section.

Prior and posterior distributions

The unknown parameters in model (2) are, μ σ_0^2 and $\boldsymbol{\beta}$. While our main concern is $\boldsymbol{\beta}$, parameters μ and σ_0^2 need to be estimated so that we can infer $\boldsymbol{\beta}$. To this end, we assign a noninformative uniform prior μ to and σ_0^2, i.e., $p(\mu) \propto 1$ and $p(\sigma_0^2) \propto 1$. Following the Bayesian LASSO model [10], we assume a three-level hierarchical model for $\boldsymbol{\beta}$. Let us denote the elements of $\boldsymbol{\beta}$ as β_i, i = 1, 2, ..., k. At the first level, β_i, i = 1, 2, ..., k follow independent normal distributions with mean zero and unknown variance $\sigma_i^2 : \beta_i \sim N(0, \sigma_i^2)$. At the second level, $\sigma_i^2, i = 1, 2, \cdots, k$, follow independent exponential distribution with a common parameter $\lambda : p(\sigma_i^2) = \lambda \exp(-\lambda \sigma_i^2)$. For a given λ, the distribution of β_i is found to be the Laplace distribution: $p(\beta_i) = \sqrt{\frac{\lambda}{2}} \exp(-\sqrt{2\lambda}|\beta_i|)$, which is known to encourage the shrinkage of β_i toward zero [4]. However, the degree of shrinkage strongly depends on the value of λ. To alleviate the problem of choosing an inappropriate value for λ, we add another level to the hierarchical model at which we assign a conjugate Gamma prior Gamma(a, b) with a shape parameter $a > 0$ and an inverse scale parameter $b > 0$ to the parameter λ. As discussed in [10], we can pre-specify appropriate values for a and b so that the Gamma prior for λ is essentially noninformative.

Let us define vector $\boldsymbol{\sigma}^2 = [\sigma_1^2, \sigma_2^2, \cdots, \sigma_k^2]$. The joint posterior distribution of all the parameters ($\mu, \sigma_0^2, \boldsymbol{\beta}, \boldsymbol{\sigma}^2, \lambda$) can be easily found [10]. In principle, MCMC simulation can be employed to draw samples from the posterior distribution for each parameter. However, since the number of parameters $2k + 3$ in our model can be very large, the fully Bayesian approach based on MCMC sampling requires a prohibitive computational cost. To avoid this problem, Xu developed an empirical Bayes method for inferring $\boldsymbol{\beta}$ [12]. Our goal here is to develop a much faster and more accurate empirical Bayes method that can easily handle tens of thousands of variables.

Maximum a posteriori estimation of variance components

Similar to the EB method of [12], our EBLASSO first estimates $\boldsymbol{\sigma}^2$, σ_0^2 and μ, and then finds the posterior distribution of $\boldsymbol{\beta}$ based on the estimated parameters. Since λ is a parameter that we do not want to estimate, we can find the prior distribution of σ_i^2 independent of λ as follows

$$p(\sigma_i^2) = \int_0^\infty p(\sigma_i^2|\lambda)p(\lambda)d\lambda = \frac{a}{b(\sigma_i^2/b + 1)^{a+1}}. \quad (3)$$

The posterior distribution of μ, $\boldsymbol{\beta}$, $\boldsymbol{\sigma}^2$ and σ_0^2 is given by

$$p(\mu, \boldsymbol{\beta}, \boldsymbol{\sigma}^2, \sigma_0^2|\mathbf{y}) \propto p(\mathbf{y}|\mu, \boldsymbol{\beta}, \sigma_0^2)p(\mu)p(\boldsymbol{\beta}|\boldsymbol{\sigma}^2)p(\boldsymbol{\sigma}^2)p(\sigma_0^2). \quad (4)$$

The marginal posterior distribution of μ, $\boldsymbol{\sigma}^2$ and σ_0^2 can then be written as

$$p(\mu, \boldsymbol{\sigma}^2, \sigma_0^2 \mid \mathbf{y}) = \int p(\mu, \boldsymbol{\beta}, \boldsymbol{\sigma}^2, \sigma_0^2 \mid \mathbf{y})d\boldsymbol{\beta}. \quad (5)$$

Let us define the precision of β_i as $\alpha_i = 1/\sigma_i^2$, i = 1, 2, ..., k and let $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, ..., \alpha_k]$. It turns out to be more convenient to estimate $\boldsymbol{\alpha}$ rather than $\boldsymbol{\sigma}^2$ as will be shown shortly. Let us collect all parameters that need to be estimated as $\boldsymbol{\theta} = (\mu, \sigma_0, \boldsymbol{\alpha})$. The log marginal posterior distribution of $\boldsymbol{\theta}$ can be found from (5) as follows

$$L(\boldsymbol{\theta}) = -\frac{1}{2}[\log|\mathbf{C}| + (\mathbf{y} - \mu)^T\mathbf{C}^{-1}(\mathbf{y} - \mu)] - \sum_{i=1}^k (a+1)\log\frac{1+b\alpha_i}{b\alpha_i} + \text{constant}, \quad (6)$$

where $\mathbf{C} = \sigma_0^2\mathbf{I} + \sum_{i=1}^k \alpha_i^{-1}\mathbf{x}_i\mathbf{x}_i^T$ is the covariance matrix of $\mathbf{y}$ with a given $\boldsymbol{\alpha}$.

Similar to the EB method [12] and the RVM [15,16], we will iteratively estimate each parameter by maximizing the log marginal posterior distribution $L(\boldsymbol{\theta})$ with the other parameters being fixed. Since it is not difficult to find the optimal μ and σ_0^2 in each iteration as shown in [12,15], we will give the expressions for μ and σ_0^2 later but focus on the estimation of $\boldsymbol{\alpha}$ now. Let us define $\mathbf{C}_{-i} = \mathbf{C} - \alpha_i^{-1}\mathbf{x}_i\mathbf{x}_i^T$ and $\widetilde{\mathbf{y}} = \mathbf{y} - \mu$. Then following the derivations in [16], we can write $L(\boldsymbol{\theta})$ in (6) as $L(\boldsymbol{\theta}) = L(\boldsymbol{\theta}_{-i}) + L(\alpha_i)$, where $L(\boldsymbol{\theta}_{-i})$ does not depend on α_i and $L(\alpha_i)$ is given by

$$L(\alpha_i) = \frac{1}{2}\left[\log\frac{\alpha_i}{\alpha_i + s_i} + \frac{q_i^2}{\alpha_i + s_i}\right] - (a+1)\log\frac{1+b\alpha_i}{b\alpha_i}, \quad (7)$$

with $s_i = \mathbf{x}_i^T\mathbf{C}_{-i}^{-1}\mathbf{x}_i$ and $q_i = \mathbf{x}_i^T\mathbf{C}_{-i}^{-1}\widetilde{\gamma}$. If we assume $a > -1.5$ and $b > 0$, we prove in the additional file that $L(\alpha_i)$ has a unique global maximum and that the optimal α_i maximizing $L(\alpha_i)$ is given by

$$\alpha_i^* = \begin{cases} r_1, & \text{if } \delta > 0 \& \Delta > 0 \& \gamma < 0 \& L(r_1) > 0 \text{ or if } \delta < 0 \\ r_2, & \text{if } \delta = 0 \& \gamma < 0, \\ \infty, & \text{otherwise,} \end{cases} \quad (8)$$

where we have defined $\delta = 2a + 2 + s_ib - bq_i^2$, $\Delta = \gamma^2 - 4\delta(2a+3)s_i^2$, $\Delta = \gamma^2 - 4\delta(2a+3)s_i^2$, $r_1 = \frac{-\gamma - \sqrt{\Delta}}{2\delta}$ and $r_2 = -(2a+3)s_i^2/\gamma$. Note that the Gamma distribution requires that $a > 0$ and $b > 0$ as we mentioned earlier. When $-1.5 < a \le 0$ as assumed here, we essentially use an improper distribution. In the additional file, we show that this improper distribution appears appropriate for getting a point estimation of α_i

given in (8). It turns out that negative values of a give one more degree of freedom to adjust the shrinkage as will be demonstrated later in the Results section. Moreover, if a = -1, the last term in the right hand side of (7) disappears. In this case, we essentially use a noninformative uniform prior. Then it is not difficult to verify that (8) gives the optimal α_i derived in [16]:

$$\alpha_i^* = \begin{cases} \dfrac{s_i^2}{q_i^2 - s_i}, & \text{if } q_i^2 > s_i \\ \infty, & \text{if } q_i^2 \leq s_i. \end{cases} \tag{9}$$

Note that α^* in (8) and (9) depends on other unknown parameters through s_i and q_i, and thus, α_i will be estimated iteratively as detailed in the EBLASSO algorithm described in the next section. Comparing with the EB method [12], our method finds each α_i (and equivalently σ_i^2) in a closed form, whereas the EB method generally needs to employ a numerical optimization algorithm to find each σ_i^2. Therefore, our method not only saves much computation but also gives more accurate estimate of each α_i. Moreover, exploiting the similar techniques used in the RVM [16], we can further increase computational efficiency as described in the ensuing section.

Fast Empirical Bayesian LASSO algorithm

Note that when $\alpha_i = \infty$, we have $\beta_i = 0$. Therefore, in each iteration, we can construct a reduced model of (2) that includes only nonzero β_is and the corresponding columns of $\mathbf{X}$. Let $\mathbf{x}_i$ be the ith column of $\mathbf{X}$. If $\alpha_i = \infty$ in the previous iteration but α_i^* is finite in the current iteration, then we add $\mathbf{x}_i$ to the model and set $\alpha_i = \alpha_i^*$; if α_i is finite in the previous iteration but $\alpha_i^* = \infty$ in the current iteration, we delete $\mathbf{x}_i$ from the model and set $\alpha_i = \infty$; if both α_i and α_i^* are finite, we retain $\mathbf{x}_i$ in the model and update α_i as $\alpha_i = \alpha_i^*$. This can be done for all is in a pre-specified order in each iteration. Alternatively, we can employ a greedy and potentially more efficient method to construct the model as described in the following. We define two iteration loops: an outer iteration loop and an inner iteration loop. In each outer iteration, we estimate μ and σ_0^2. In the inner iterations, assuming μ and σ_0^2 are known and fixed, we estimate each α_i and construct the model. Specifically, in each inner iteration, we first calculate each α_i^* from (8) and then find $j = \arg_i \max\{\Delta L(\alpha_i^*) = L(\alpha_i^*) - L(\alpha_i), \forall i\}$, where α_i stands for the value of α_i obtained in the previous inner iteration. This step basically identifies the $\mathbf{x}_j$ that gives the largest increase in the log posterior distribution. Then we add, delete or retain $\mathbf{x}_j$ as described early. The inner iterations can run until a local convergence criterion is satisfied. Let vector $\tilde{\boldsymbol{\alpha}}$ contain all finite α_is, vector $\tilde{\boldsymbol{\beta}}$ consist of the corresponding β_is and matrix $\tilde{\mathbf{X}}$

contain the corresponding columns of $\mathbf{X}$. Then we essentially construct the following reduced model:

$$\mathbf{y} = \mu + \tilde{\mathbf{X}}\tilde{\boldsymbol{\beta}} + \mathbf{e}, \tag{10}$$

where the number of columns of $\tilde{\mathbf{X}}$, k_r, is typically much less than the number of rows, n. This property will be used to reduce computation.

To calculate α_i^* in (8), we need to first calculate s_i and q_i which requires $\mathbf{C}_{-i}^{-1}$. Since $\mathbf{C}_{-i}$ is different for different i, it may need large computation to calculate all $\mathbf{C}_{-i}^{-1}$. However, it was shown in [16] that we can calculate s_i and q_i as follows

$$s_i = \frac{\alpha_i S_i}{\alpha_i - S_i}$$
$$q_i = \frac{\alpha_i Q_i}{\alpha_i - S_i}, \tag{11}$$

where $S_i = \mathbf{x}_i^T \mathbf{C}^{-1} \mathbf{x}_i$ and $Q_i = \mathbf{x}_i^T \mathbf{C}^{-1} \tilde{\mathbf{y}}$. This requires only one matrix inversion in each iteration for calculating all s_i and q_i, i = 1, ..., k. However, since $\mathbf{C}$ is a relatively large matrix of size $n \times n$, direct calculation of $\mathbf{C}^{-1}$ may still require large computation. To avoid this problem, we can use the Woodbury matrix identity to derive an expression for $\mathbf{C}^{-1}$:

$$\mathbf{C}^{-1} = \sigma_0^{-2}\mathbf{I} - \sigma_0^{-4}\tilde{\mathbf{X}}\boldsymbol{\Sigma}\tilde{\mathbf{X}}^T, \tag{12}$$

where

$$\boldsymbol{\Sigma} = (\mathbf{A} + \sigma_0^{-2}\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1} \tag{13}$$

with $\mathbf{A}$ = diag $(\tilde{\boldsymbol{\alpha}})$. The size of matrix $\boldsymbol{\Sigma}$ is $k_r \times k_r$ which is typically much smaller than the size of $\mathbf{C}$. Since inverting a matrix of $N \times N$ using an efficient method such as QR decomposition needs computation of $O(N^3)$, calculating $\boldsymbol{\Sigma}$ requires much less computation than directly inverting $\mathbf{C}$. Using (12), we can calculate S_i and Q_i as follows:

$$S_i = \sigma_0^{-2}\mathbf{x}_i^T\mathbf{x}_i - \sigma_0^{-4}\mathbf{x}_i^T\tilde{\mathbf{X}}\boldsymbol{\Sigma}\tilde{\mathbf{X}}^T\mathbf{x}_i$$
$$Q_i = \sigma_0^{-2}\mathbf{x}_i^T\tilde{\mathbf{y}} - \sigma_0^{-4}\mathbf{x}_i^T\tilde{\mathbf{X}}\boldsymbol{\Sigma}\tilde{\mathbf{X}}^T\tilde{\mathbf{y}}. \tag{14}$$

So far we have derived the method for efficiently estimating α. Other two unknown parameters μ and σ_0^2 can be obtained by setting $\frac{\partial L(\boldsymbol{\theta})}{\partial \mu} = 0$ and $\frac{\partial L(\boldsymbol{\theta})}{\partial \sigma_0^2} = 0$. This gives

$$\mu = \mathbf{1}^T\mathbf{C}^{-1}\mathbf{y}/(\mathbf{1}^T\mathbf{C}^{-1}\mathbf{1}), \tag{15}$$

where $\mathbf{1}$ is a vector whose elements are all 1, and [15]

$$\sigma_0^2 = \frac{|\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\mathbf{u}|^2}{n - k_r + \sum_{i=1}^{k_r} \tilde{\alpha}_i \Sigma_{ii}}, \tag{16}$$

where Σ_{ii} is the ith diagonal element of Σ and

$$\mathbf{u} = \sigma_0^{-2} \Sigma \tilde{\mathbf{X}}^T \tilde{\mathbf{y}}. \tag{17}$$

After we get estimates of μ, σ_0^2 and $\boldsymbol{\alpha}$, we finalize the model (10), where the prior distribution of $\tilde{\beta}_i$ is now $N(0, \tilde{\alpha}_i^{-1})$, $i = 1, 2, \cdots, k_r$. For those $\mathbf{x}_i$s not in the model, we can declare that they do not affect the quantitative trait because their regression coefficient is zero. For those $\mathbf{x}_i$s in the matrix $\tilde{\mathbf{X}}$, the posterior distribution of their regression coefficients $\tilde{\boldsymbol{\beta}}$ is Gaussian with covariance Σ in (13) and mean $\mathbf{u}$ in (17) [15]. We can then use the z-score or more conservative t-statistics to test if $\tilde{\beta}_i \neq 0$ at certain significance level. In this paper, the posterior mean $\mathbf{u}_j$ of the jth effect is reported as the empirical Bayes estimate of β_j, denoted by $\hat{\beta}_j$, and the posterior standard deviation, $s_{\hat{\beta}_j} = (\Sigma_{jj})^{1/2}$, is used as the standard error of $\hat{\beta}_j$.

We now summarize our fast EBLASSO algorithm as follows.

Algorithm 1 (EBLASSO algorithm)

1. Initialize parameters: choose $a > -1.5$ and $b > 0$, calculate $\mu = \mathbf{1}^T \mathbf{y}/n$, $\tilde{\mathbf{y}} = \mathbf{y} - \mu$ and set σ_0^2 to be a small number, e.g., $0.1 \times \tilde{\mathbf{y}}^T \tilde{\mathbf{y}}/n$.

2. Initialize the model: Find $j = \arg_i \max\{|\mathbf{x}_i^T \tilde{\mathbf{y}}|, \forall i\}$, and calculate α_j from (9), set all other α_is to be ∞ and $\tilde{\mathbf{X}} = \mathbf{x}_j$.

3. Calculate Σ from (13), S_i and Q_i, $\forall i$, from (14).

4. Update the model

   while the local convergence criterion is not satisfied

     Calculate q_i and s_i from (11), $\forall i$.

     Calculate α_i^* from (8), $\forall i$.

   Find $j = \arg_i \max\{\Delta L(\alpha_i^*) = L(\alpha_i^*) - L(\alpha_i^{(n)})\}$.

   if $\alpha_j^* = \infty$

     if $\mathbf{x}_j$ is in the model, delete it and update Σ, S_i and Q_i, $\forall i$.

     else

     if $\mathbf{x}_j$ is in the model, set $\alpha_j = \alpha_j^*$ and update Σ, S_i and Q_i, $\forall i$.

     if $\mathbf{x}_j$ is not in the model, add it, set $\alpha_j = \alpha_j^*$ and update Σ, S_i and Q_i, $\forall i$.

     end if

   end while

5. Update the residual variance σ_0^2 using (16).

6. Calculate Σ from (13) and $\mathbf{C}^{-1}$ from (12).

7. Update the fixed effects μ using (15) and update $\tilde{\mathbf{y}} = \mathbf{y} - \mu$.

8. Calculate S_i and Q_i from (14).

9. If the global convergence criterion is not satisfied, go to step 4.

10. Find the posterior mean of $\tilde{\beta}_i$ from (17) and the posterior variance Σ_{ii} from (13).

11. Use t-statistics to test if $\tilde{\beta}_i \neq 0$.

The parameters a and b can be set to be a small number (e.g., $a = b = 0.01$) so that the Gamma prior distribution is essentially noninformative [10]. Alternatively, we can use the predicted error (PE) obtained from cross-validation [4] to choose the values of a and b. As done in [16], the initial value of σ_0^2 is chosen in step 1 to be a small number and the initial model is selected in step 2 to have a single effect that corresponds to the maximum $L(\alpha_i)$ with $a = -1$. The outer iteration loop consists of steps 4-9, while the inner iteration loop is step 4, where we use the greedy method described earlier to update the model. In step 4, we use the method given in the Appendix of [16] to efficiently update Σ, S_i and Q_i after a $\mathbf{x}_j$ is added to or deleted from the model or after α_j is updated. The local convergence criterion can be defined as the simultaneous satisfaction of the following three conditions: 1) no effect can be added to or deleted from the model, 2) the change of $L(\boldsymbol{\theta})$ between two consecutive inner iterations is smaller than a pre-specified small value, and 3) the change of $\tilde{\boldsymbol{\alpha}}$ between two consecutive inner iterations is less than a pre-specified value. The global convergence criterion can be defined as the simultaneous satisfaction of the following two conditions: the change of $L(\boldsymbol{\theta})$ between two consecutive outer iterations is smaller than a pre-specified small value, and 2) the total change in μ, σ_0^2 and $\tilde{\boldsymbol{\alpha}}$ between two consecutive outer iterations is less than a pre-specified value. A Matlab program has been developed to implement the algorithm; and a more efficient C++ program is under development.

Results

Simulation study

We simulated a single large chromosome of 2400 centi-Morgan (cM) long covered by evenly spaced $q = 481$ markers with a marker interval of 5 cM. The simulated population was an F_2 family derived from the cross of two inbred lines with a sample size $n = 1000$. The genotype indicator variable for individual i at marker k is defines as $X_{ik} = 1, 0, -1$ for the three genotypes, A_1A_1, A_1A_2 and A_2A_2, respectively. Twenty markers are QTLs with main effects and 20 out of the $\binom{481}{2} = 115, 440$ marker pairs have interaction effects. The locations and effects of the markers and maker pairs are shown in Table 1. Environmental effects and $G \times E$ effects were not simulated. The true population mean is $\mu = 100$ and the residual variance is $\sigma_0^2 = 10$.

The total phenotypic variance for the trait can be written as

Table 1 True and estimated QTL effects for the simulated data with main and epistatic effects

Markers[a] (i, j)	Position (cM, cM)	True[b] $\beta(h^2)$	EBLASSO[c] $\hat{\beta}(s_{\hat{\beta}})$	EB[c] $\hat{\beta}(s_{\hat{\beta}})$
(11,11)	(50,50)	4.47(0.0975)	4.5801(0.1612)	4.8593(0.2075)
(26,26)	(125,125)	3.16(0.0524)	3.0768(0.1576)	3.3221(0.2035)
(42,42)	(205,205)	-2.24(0.0250)	-2.3169(0.1796)	-2.2769(0.2262)
(48,48)	(235,235)	-1.58(0.0128)	-1.3171(0.1720)	-1.3634(0.2205)
(72,72)	(355,355)	2.24(0.0247)	-	1.6537(0.4277)
(73,73)	(360,360)	3.16(0.0506)	5.1247(0.1555)	3.8771(0.4219)
(123,123)	(610,610)	1.10(0.0062)	-	1.5168(0.2432)
(127,127)	(630,630)	-1.10(0.0063)	-	-1.1834(0.2460)
(161,161)	(800,800)	0.77(0.0030)	-	-
(181,181)	(900,900)	1.73(0.0152)	-	-
(182,182)	(905,905)	3.81(0.0725)	5.6744(0.2400)	5.5127(0.2894)
(185,185)	(920,920)	2.25(0.0263)	1.7123(0.2327)	1.7070(0.2858)
(221,221)	(1100,1100)	-1.30(0.0088)	-1.4276(0.1506)	-1.0867(0.1956)[d]
(243,243)	(1210,1210)	-1.00(0.0051)	-0.8603(0.1486)	-
(262,262)	(1305,1305)	-2.24(0.0245)	-2.2539(0.1826)	-1.6078(0.2417)
(268,268)	(1335,1335)	1.58(0.0120)	2.4264(0.2040)	2.1736(0.2509)
(270,270)	(1345,1345)	1.00(0.0049)	-	-
(274,274)	(1365,1365)	-1.73(0.0147)	-1.4114(0.1800)	-1.4935(0.2254)
(361,361)	(1800,1800)	0.71(0.0026)	0.7856(0.1457)	0.6520(0.1859)[d]
(461,461)	(2300,2300)	0.89(0.0040)	-	-
(5,6)	(20,25)	2.24(0.0230)	1.7839(0.1654)	1.5752(0.2886)
(6,39)	(25,190)	2.25(0.0128)	1.9691(0.2168)	-
(42,220)	(205,1095)	4.47(0.0511)	4.3836(0.2198)	4.6414(0.3394)
(75,431)	(370,2150)	0.77(0.0014)	1.1360(0.2124)[d]	-
(81,200)	(400,995)	-2.24(0.0128)	-2.4190(0.2460)	-
(82,193)	(405,960)	1.58(0.0063)	1.6345(0.2442)	-
(87,164)	(430,815)	3.16(0.0235)	2.9263(0.2254)	1.7059(0.3319)[d]
(87,322)	(430,1605)	3.81(0.0342)	4.1019(0.2274)	3.7040(0.3632)
(92,395)	(455,1970)	1.73(0.0081)	1.5714(0.2065)[d]	-
(104,328)	(515,1635)	1.00(0.0024)	0.8081(0.1979)[d]	-
(118,278)	(585,1385)	-2.24(0.0120)	-2.0796(0.2221)	-2.2590(0.3460)
(150,269)	(745,1340)	1.10(0.0028)	1.0740(0.2142)	-
(237,313)	(1180,1560)	0.71(0.0014)	-	-
(246,470)	(1225,2345)	-1.10(0.0032)	-1.2381(0.2114)[d]	-
(323,464)	(1610,2315)	0.89(0.0020)	-	-
(328,404)	(1635,2015)	-1.73(0.0079)	-2.3036(0.2123)	-1.9428(0.3330)
(342,420)	(1705,2095)	-1.30(0.0041)	-1.3886(0.2121)[d]	-
(344,407)	(1715,2030)	-1.00(0.0025)	-	-
(373,400)	(1860,1995)	-1.58(0.0070)	-1.4732(0.2028)	-
(431,439)	(2150,2190)	3.16(0.0278)	2.6700(0.2121)	2.2454(0.3366)[d]
μ		100	100.70	100.59
σ_0^2		10	11.76	0.25
CPU time			3.4 mins	249 hrs

[a]When $i = j$, the QTL is a main effect; otherwise, it is an epistatic effect.

[b]The true value of a QTL effect is denoted by β and the proportion of variance contributed by the QTL is denoted by h^2.

[c]The estimated QTL effect is denoted by $\hat{\beta}$ and the standard error is denoted by $s_{\hat{\beta}}$. The EBLASSO algorithm used hyperparameters $a = b = 0.1$ and the EB algorithm used hyperparameters $\tau = -1$ and $\omega = 0.001$.

[d]The estimated QTL effect was obtained from a neighboring marker (≤ 20 cM away) rather than from the maker with the true effect.

$$\sigma_\gamma^2 = \sigma^2 + \sum_{j=1}^{q} \sum_{j'=1}^{q} \beta_j \beta_{j'} \text{cov}(x_j, x_{j'}), \qquad (18)$$

where $\text{cov}(x_j, x_{j'})$ is the covariance between X_j and $X_{j'}$ if $j \neq j'$ or the variance of X_j if $j = j'$, which can be estimated from the data. The total phenotypic variance was calculated from (18) to be $\sigma_\gamma^2 = 98.67$ and the total genetic variance contributed by the main and epistatic effects of the markers was calculated from the second term of the right hand side of (18) to be 88.67. If we ignore the contributions from the covariance terms which are relatively small, the proportions of the phenotypic variance explained by a particular QTL effect j can be approximated by

$$h_j^2 = \frac{\beta_j^2 \text{var}(x_j)}{\sigma_\gamma^2}, \qquad (19)$$

where $\text{var}(x_j)$ is the variance of X_j. In the simulated data, the proportion of contribution from an individual QTL varied from 0.30% to 9.75%, whereas the proportion of contribution from a pair of QTLs ranged from 0.26% to 7.25%, as shown in Table 1. Some of the markers had only main or epistatic effect, while the others had both main and epistatic effects. The QTL model contained a total of $1 + 481 + \binom{481}{2} = 115,922$ possible effects, a number about 115 times of the sample size.

The data were analyzed in Matlab on a personal computer (PC) using the EBLASSO algorithm, the EB method, the RVM and the LASSO. The Matlab program SPARSEBAYES for the RVM was downloaded from http://www.miketipping.com. We translated the original SAS program for the EB method [12] into Matlab, and slightly modified the program to avoid possible memory overflow due to the large number of possible effects. We also got the program glmnet [19] that is a very efficient program implementing the LASSO and other related algorithms. The PC version of glmnet uses Matlab to initialize and call the core LASSO algorithm that is implemented efficiently with Fortran code.

We used the PE [4] obtained from ten-fold cross validation to select the values of hyperparameters a and b in our EBLASSO algorithm. Ideally, we should test a large set of values for $a \geq -1$ and $b > 0$, but this may be time consuming. Therefore, we first ran cross-validation for the following values: $a = b = 0.001, 0.01, 0.05, 0.1, 0.5, 1$; the degree of shrinkage generally decreases along this path. Table 2 lists the PEs and the standard errors of the PEs for different values of a and b. It is seen that the PE for $a = b = 0.1$ is the smallest, although it is close to the PEs for $a = b = 0.05$ and 0.5 but relatively smaller than PEs for the other values of a and b. To see if $a = b = 0.1$ is the best set of values, we further ran

Table 2 Summary of results for the simulated data with main and epistatic effects

Algorithm	Parameters[◦]	PE ± STE*	Number of effects[†‡]	$(\hat{\sigma}_0^2)^{\dagger}$	CPU time (mins)
	(0.001, 0.001)	16.49 ± 0.8908	25/0	13.5	3.4
	(0.01, 0.01)	15.95 ± 0.7477	28/0	12.46	3.4
	(0.05, 0.05)	15.89 ± 0.7498	30/0	11.72	3.4
	(0.1, 0.1)	15.81 ± 0.8359	30/0	11.72	3.4
EBLASSO	(0.5, 0.5)	15.86 ± 0.7717	31/0	11.57	3.4
	(1, 1)	16.07 ± 0.7203	29/0	12.31	3.4
	(0.5, 0.1)	16.14 ± 0.8557	28/0	12.5	3.4
	(-0.01, 0.1)	15.92 ± 1.0161	32/1	11.31	3.4
	(-1, 0.0001)	-	14/1	21.22	2,760.0
EB	(-1, 0.0005)	-	13/1	12.15	4,140.0
	(-1, 0.001)	-	22/1	0.25	14,940.0
	(-1, 0.01)	-	8/0	0.01	2,760.0

[◦] Parameters are (*a, b*) for the EBLASSO, and (*τ, ω*) for the EB.
*The average PE and the standard error were obtained from ten-fold cross validation.
[†]The number of effects and residual variance were obtained using all 1000 samples not from cross validation.
[‡]The first number is the number of true positive effects; the second number is number of false positive effects. All the effects counted have a p-value ≤ 0.05.

cross-validation with *b* = 0.1 and *a* = 0.5 or -0.01. For a fixed *b*, the degree of shrinkage decreases when *a* decreases. It is seen from Table 2 that the PE for *b* = 0.1 and *a* = 0.5 or -0.01 is greater than that for *a* = *b* = 0.1. Therefore, cross-validation gave *a* = *b* = 0.1 as the best set of values. Table 2 also lists the number of effects detected with different values of *a* and *b*. All 30 effects detected with *a* = *b* = 0.1 are presented in Table 1 and shown in Figures 1 and 2.

To test the performance of the EB method, we ran the program with the following parameter values: *τ* = -1, *ω* = 0.0001, 0.0005, 0.001 and 0.01, which yielded 14, 13, 22, and 8 true simulated effects, respectively, and 1, 1, 1, and 0 false effects, respectively, as shown in Table 2. Cross-validation was not done because it was too time-consuming, and thus, the optimal values for the parameters could not be determined. Nevertheless, we listed 22 true positive effects estimated with *τ* = -1 and *ω* = 0.001 in Table 1, which reflects the best performance of the EB method with the set of parameters values tested. We also plotted these 22 effects in Figures 3 and 4. Comparing the effects detected by EBLASSO and EB methods shown in Table 1 and in Figures 1, 2, 3 and 4, the EBLASSO method detected 13 (17) true main (epistatic) effects, whereas the EB method detected 15 (7) true main (epistatic) effects. Overall, the EBLASSO detected 8 more true effects than the EB method without any false positive effects, whereas the EB method gave one false positive effect. We would like to
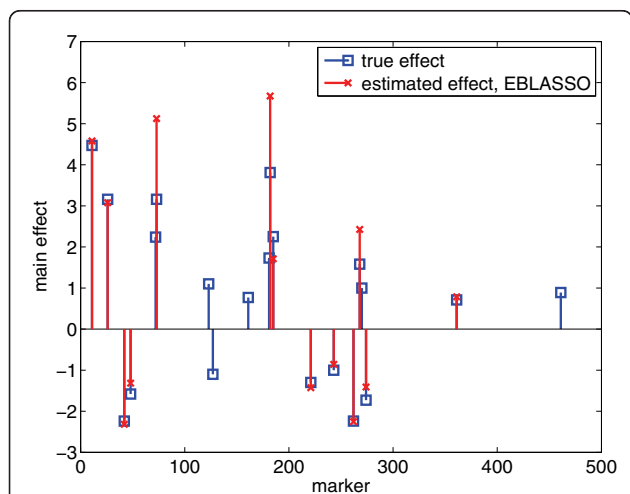

Figure 1 Main effects estimated with the EBLASSO for the simulated data with main and epistatic effects.
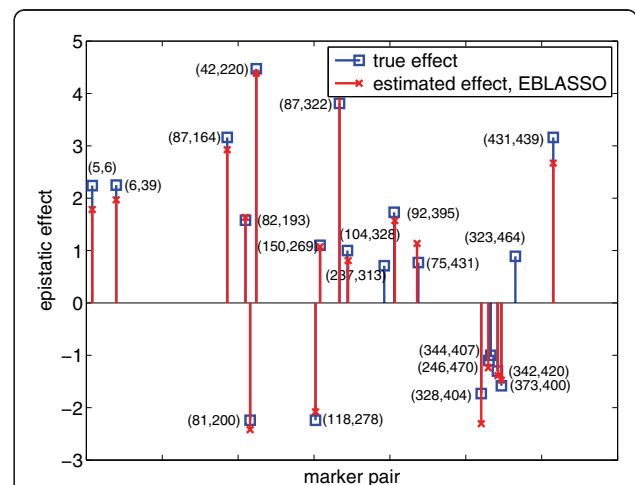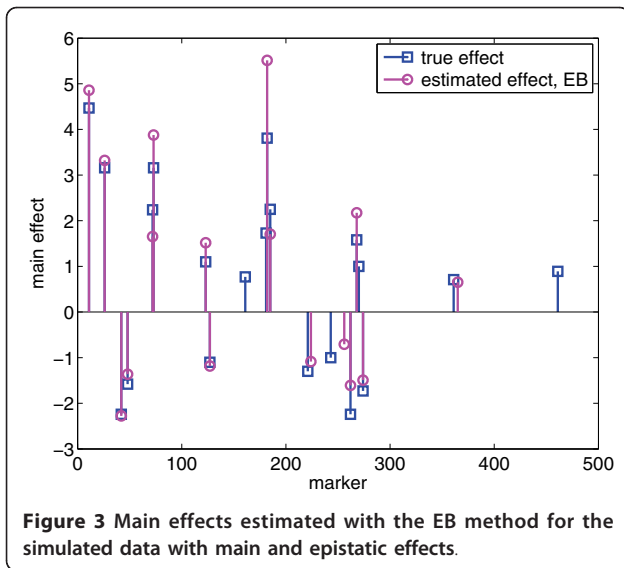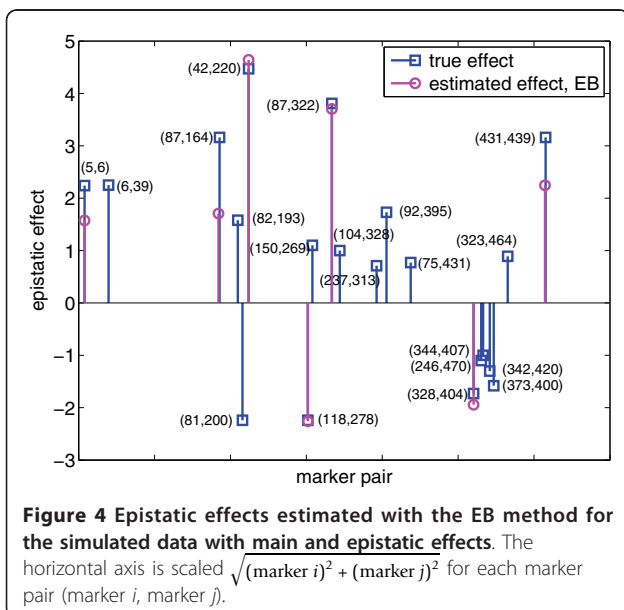

Figure 2 Epistatic effects estimated with the EBLASSO for the simulated data with main and epistatic effects. The horizontal axis is scaled as $\sqrt{(\text{marker } i)^2 + (\text{marker } j)^2}$ for each marker pair (marker *i*, marker *j*).

Figure 3 Main effects estimated with the EB method for the simulated data with main and epistatic effects.

emphasize here that the EB method may detect less number of true effects in practice because as we mentioned earlier it is too time-consuming to choose the optimal values for the parameters τ and ω. To see if EBLASSO could estimate QTL effects robustly, we simulated three replicates of the data: each replicate consists of 1000 individuals whose genotypes at 481 markers were independently generated and whose phenotypes were calculated from (2) with **e** independently generated from Gaussian random variables with zero mean and covariance 10**I**. We performed cross-validation and determined the best values of a and b for each replicate. Using these values, we ran the EBLASSO and identified 35, 38, 34 true positive effects and 4, 6, 2 false

Figure 4 Epistatic effects estimated with the EB method for the simulated data with main and epistatic effects. The horizontal axis is scaled $\sqrt{(\text{marker } i)^2 + (\text{marker } j)^2}$ for each marker pair (marker i, marker j).

positive effects, respectively, for three replicates. These results showed that the EBLASSO could detect QTL effects robustly.

The EBLASSO took about 3.4 minutes of CPU time for each set of values of a and b listed in Table 2, whereas the EB took 249 hours of CPU time for $\tau = -1$ and $\omega = 0.001$ and about 46, 69, 46 hours for $\tau = -1$ and $\omega = 0.0001, 0.0005, 0.01$, respectively. This simulation study showed that the EBLASSO method not only can detect more effects, but also offers a huge advantage in terms of computational time. Note that all simulations were done in Matlab. It is expected that the EBLASSO algorithm will be even faster, after its implementation in C++ is completed.

We wished to test the performance of the RVM and the LASSO on the simulated data. To this end, we replaced the inner iteration in our EBLASSO algorithm with the program SPARSEBAYES that implemented the RVM. Although we carefully modified SPARSEBAYES to avoid possible memory overflow due to the large number of possible effects, the program ran out of memory after one or two outer iterations. Hence, we did not get any results from the RVM method. Considering the QTL model in (2), the LASSO tries to estimate μ and the QTL effects $\boldsymbol{\beta}$ as follows

$$\min_{\mu, \boldsymbol{\beta}} \left[\frac{1}{2n} (\mathbf{y} - \mu - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mu - \mathbf{X}\boldsymbol{\beta}) + \lambda \sum_{i=1}^{k} |\beta_i| \right], \quad (20)$$

where λ is a positive constant that can be determined with cross-validation [4]. We tried to run the program glmnet [19] with the simulated data. However, glmnet could not handle the big design matrix **X** of 1, 000 × 115, 922 in our QTL model, and we did not get any results from glmnet.

In order to compare the performance of our EBLASSO algorithm with that of the RVM and the LASSO, we simulated a new set of data by deleting the 20 epistatic effects in the previous set of simulated data, and then used a QTL model that only contained 481 possible main effects to estimate QTL effects, i.e., the design matrix **X** in (2) was 1000 × 481. The small number of possible effects was chosen to avoid the memory overflow problem of the RVM and glmnet. The results of the EBLASSO, EB, RVM and LASSO for this data set are summarized in Table 3. To choose the values of a and b for the EBLASSO, we ran ten-fold cross validation with the following parameters: $a = b = 0.001, 0.01, 0.05, 0.1, 0.5, 1$. Since $a = b = 0.1$ yielded the smallest PE, we further performed cross-validation with $b = 1$, and $a = -0.95, -0.75, -0.5, 0.5$. This gave the smallest PE at $a = -0.75$ and $b = 0.1$. Finally, we used $a = -0.75$ and $b = 0.1$ to run the EBLASSO algorithm on the whole data set, which identified all 20 true effects but also 5 false

Table 3 Summary of results for the simulated data with only main effects

Algorithm	Parameters°	PE ± STE*	Number of effects[†‡]	$(\hat{\sigma}_0^2)^{\dagger}$	CPU time (sec)
	(0.001, 0.001)	11.52 ± 0.5677	14/0	11.1	1.2
	(0.01, 0.01)	11.52 ± 0.578	16/0	10.53	1.3
	(0.05, 0.05)	11.36 ± 0.6088	17/0	10.32	1.1
	(0.1, 0.1)	11.23 ± 0.5571	17/0	10.32	1.1
	(0.5, 0.5)	11.32 ± 0.5937	17/0	10.34	1.1
EBLASSO	(1, 1)	11.4 ± 0.5929	16/0	10.64	1.1
	(0.5, 0.1)	11.57 ± 0.5593	15/0	10.83	1.3
	(-0.5, 0.1)	10.87 ± 0.5599	17/0	10.31	1.6
	(-0.75, 0.1)	10.78 ± 0.5646	20/5	9.52	1.5
	(-0.95, 0.1)	11.09 ± 0.5045	22/20	8.71	1.4
	(-1, 0.0001)	17.73 ± 2.0244	9/0	16.07	1491.9
EB	(-1, 0.0005)	15.81 ± 2.5732	16/0	11.66	1676.0
	(-1, 0.001)	12.21 ± 1.7635	17/2	10.65	1657.9
	(-1, 0.01)	10.69 ± 0.9903	19/4	9.05	1954.9
	(-1, 0.1)	11.63 ± 0.5743	20/20	7.29	2222.7
RVM	-	-	20/42	7.81	268.7
	0.1347	10.77 ± 0.4583	16/27	9.47	0.7
	0.0850	10.52 ± 0.4442	20/49	8.89	0.7
LASSO	0.0675	10.50 ± 0.5248	20/48	8.63	0.7
	0.0536	10.52 ± 0.4382	19/35	8.28	0.7
	0.0338	10.59 ± 0.4434	17/2	7.35	0.7

° Parameters are (*a, b*) for the EBLASSO, (*τ, ω*) for the EB and *λ* for the LASSO.

*The average PE and the standard error were obtained from ten-fold cross validation.

[†]The number of detected effects and residual variance were obtained using all 1000 samples not from cross validation.

[‡]The first number is the number of true positive effects; the second number is number of false positive effects. All the effects counted have a p-value ≤ 0.05.

positive effects. In fact, since the EBLASSO program ran very fast on this data set, we did ten-fold cross validation for 165 sets of values for *a* and *b* obtained by combining the following values: *a* = -1, -0.95, -0.85, -0.75, -0.5, -0.1, -0.05, -0.01, -0.001, 0.001, 0.01, 0.05, 0.1, 0.5, 1 and *b* = 0.001 0.01 0.05 0.1, 0.5, 1, 2.5, 5, 10, 15, 20. The smallest average PE and standard error were 10.49 ± 0.5444 achieved at *a* = 0.05 and *b* = 10. With this set of values, the EBLASSO identified all 20 true effects and also 6 false positive effects, which is similar to that identified at *a* = -0.75 and *b* = 0.1. This study showed that the two-step cross-validation approach, where cross-validation was run with *a* = *b* = 0.001, 0.01, 0.05, 0.1, 0.5, 1 and then with a fixed *b* but a varying *a*, performed well and could save much time. We further simulated three replicates of the data in the same way described earlier for the model with epistatic effects. We then repeated the two-step cross-validation for each replicate. Using the values of *a* and *b* determined in cross-validation, the EBLASSO detected 20, 19, 19 true effects, and 4, 1 and 2 false positive effects, respectively, for three replicates. This again showed that our EBLASSO could estimate the effects robustly.

The optimal values for the parameters of the EB method were *τ* = -1 and *ω* = 0.01, since they gave the smallest PE in cross-validation as listed in Table 3. With *τ* = -1 and *ω* = 0.01, the EB method detected 19 true effects and 4 false positive effects. The RVM detected all 20 true effects as the EBLASSO did, but it also output a large number of 42 false positive effects. This result is consistent the observation [13] that the uniform prior distribution used in the RVM usually yields many false positive effects. To choose the optimal value of *λ* for the LASSO, we ran ten-fold cross validation starting from *λ* = 4.9725 (which gave only one nonzero effect) and then decreasing *λ* to 0.0025 with a step size of 0.0768 on the logarithmic scale (Δ ln(*λ*) = 0.0768). The smallest PE was achieved at *λ* = 0.0675. We then used this value to run glmnet on the whole data set, which yielded 97 nonzero effects. For each of these nonzero effects, we calculated their standard error using equation (7) in [4], and then calculated the p-value of each nonzero effect. This gave 20 true effects and 48 false positive effects with a p-value less than 0.05. Comparing the number of effects detected by the EBLASSO, EB, RVM and LASSO, the EBLASSO offered the best performance because it detected all true effects and a very small number of false positive effects.

It is seen from Table 3 that the EBLASSO and the LASSO took much less time than the EB method and

the RVM on analyzing this data set. It is expected that the EBLASSO is much faster than the EB method because as we discussed earlier the EB needs a numerical optimization procedure. The RVM and EBLASSO generally should have similar speed because two algorithms use the similar technique to estimate effects. However, when applying to the same data set, the RVM often yields a model with much more nonzero effects than the EBLASSO as is the case here, because the RVM does not provide sufficient degree of shrinkage. Due to this reason, the RVM algorithm requires more time than the EBLASSO. The LASSO took slightly less CPU time than the EBLASSO in this example. However, we would emphasize that the LASSO was implemented with Fortran but our EBLASSO was implemented with Matlab. The speed of EBLASSO is expected to increase significantly once it is implemented in C/C++.

Real data analysis

This dataset was obtained from [20]. This dataset consists of n = 150 double haploids (DH) derived from the cross of two spring barley varieties Morex and Steptoe. The total number of markers was q = 495 distributed along seven pairs of chromosomes of the barley genome, covering 206 cM of the barley genome. The phenotype was the spot blotch resistance measured as the lesion size on the leaves of barley seedlings. Note that spot blotch is a fungus named *Cochliobolus sativus*. This dataset was used as an example for the application of the EBLASSO method. Genotype of the markers were encoded as +1 for genotype A (the Morex parent), -1 for genotype B (the Steptoe parent), and 0 for missing genotype. Ideally, the missing genotypes should be imputed from known genotypes of neighboring markers. For simplicity, we replaced the missing genotypes with 0 in order to use the phenotypes of the individuals with missing genotypes. The total missing genotypes only account for about 4.2% of all the genotypes. Including the population mean, the main and the pair-wise epistatic effects, the total number of model effects was $1 + 495 + \binom{495}{2}$ = 122, 761, about 818 times as large as the sample size.

Table 4 gives the average PE and the standard error obtained from 5-fold cross validation, the residual variance and the number of effects detected by the EBLASSO method for different values of a and b. It is seen that the PEs for a = b = 0.001, 0.01.0.05 are almost the same but are smaller than the PEs for other larger a and b. However, when a = b = 0.001 or 0.01, only one or two effects were detected. When a = b = 0.1, 0.5 or 1, the residual variance is very small, implying that the model is likely over-fitted. Specifically, the number of columns of matrix $\tilde{X}$ in the model (10) is equal to the

Table 4 Summary of the results of the EBLASSO algorithm for the real data

$a = b$	PE ± STE*	Number of effects[†‡]	$(\hat{\sigma}_0^2)^{\dagger}$
0.001	0.70 ± 0.21	1/1/1	0.6706
0.01	0.79 ± 0.31	2/2/2	0.5996
0.05	0.70 ± 0.21	11/11/11	0.2699
0.07	0.96 ± 0.30	10/15/15	0.2104
0.1	1.20 ± 0.18	13/128/132	2.59E-06
0.5	1.21 ± 0.09	9/112/122	2.59E-06
1	1.25 ± 0.17	8/115/132	2.59E-06

*The average PE and the standard error were obtained from five-fold cross validation.
[†]The number of effects and residual variance were obtained using all 150 samples not from cross validation.
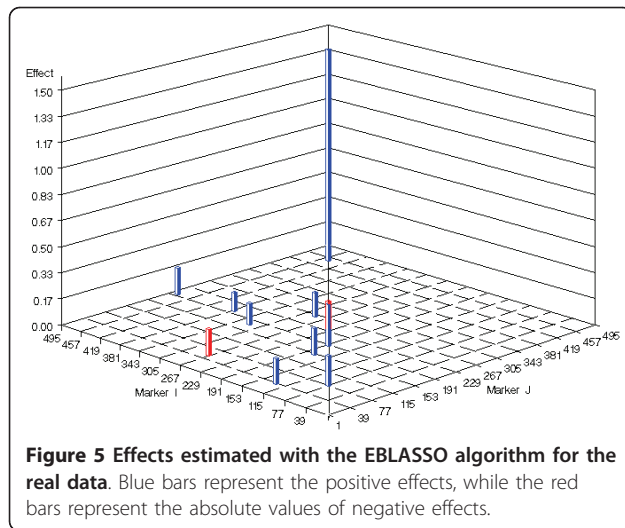[‡]The first number is the number of effects with a p-value ≤ 0.05 and proportion of variance ≥ 0.5%; the second number is the number of effects with a p-value ≤ 0.05; the third number is the total number of non-zero effects reported by the program.

total number nonzero effects, which is more than 120 for a = b = 0.1, 0.5 or 1 as indicated in Table 4. Hence, since the number of samples (150) is relative small, y - μ can be almost completely in the column space of $\tilde{X}$, which results in very small residual variance. Based on these observations, it seems that a = b = 0.05 gives reasonable results, because the PE is among the smallest and the residual variance is relatively but not unreasonably small. Nevertheless, in order to estimate effects more reliably, we searched over all effects detected with a = b = 0.05, 0.07, 0.1, 0.5 or 1, and found eight effects are detected with all these values. Markers or marker pairs of these eight effects and their values estimated with a = b = 0.05 were listed in Table 5. All 11 effects estimated with a = b = 0.05 were also plotted in Figure 5. As seen from Table 5, one single QTL at marker 446 contributes most of the phenotypic variance

Table 5 Eight effects estimated with the EBLASSO algorithm for the real data

Markers	$\hat{\beta}_j(s_{\hat{\beta}_j})$	$\hat{h}_j^2$
(446,446)	1.4173(0.0432)	0.7639
(187,187)	0.2624(0.0421)	0.0262
(77,77)	0.1881(0.0413)	0.0132
(238,238)	-0.1742(0.0427)	0.0101
(197,483)	0.1748(0.0405)	0.0117
(37,130)	0.1557(0.0422)	0.0085
(53,270)	-0.1649(0.0452)	0.0081
(149,175)	0.1697(0.0464)	0.0078
$\hat{\mu}$	5.7037	
$\hat{\sigma}_0^2$	0.2699	
$\hat{h}^2$		0.8495

Eights effects were detected with all of the following parameters: a = b = 0.05, 0.07, 0.1, 0.5, 1; $\hat{\beta}_j$, $s_{\hat{\beta}_j}$ and $\hat{h}_j^2$ listed here were obtained with a = b = 0.05.

Figure 5 Effects estimated with the EBLASSO algorithm for the real data. Blue bars represent the positive effects, while the red bars represent the absolute values of negative effects.

(about 76%), while the other QTL effects contribute from about 0.8% to 2.6% of the variance. We also analyzed the data with the EB method. One effect at marker 446 was detected with a p-value < 0.05 when $\tau = -1$, $\omega = 0.0001$ or 0.0005, no effect was detected with p-value ≤ 0.05 when $\tau = -1$, $\omega = 0.001$ or 0.01. The CPU time of the EB was about 46 minutes for each set of values of τ and ω tested; whereas the CPU time of the EBLASSO method was about 3 minutes for $a = b = 0.05$.

Discussion

Xu [12] compared several methods for multiple QTL mapping including the EB [12], LASSO [4], penalized likelihood (PENAL) [21] and stochastic search variable selection (SSVS) [22,23] methods. The SSVS method is much slower than the EB method; whereas LASSO and PENAL methods are faster than the EB method. Although we did not directly compare the speed of our EBLASSO with that of the PENAL method, based on all comparisons with the EB method in [12], we observed that the EBLASSO method is faster than PENAL methods. Direct comparison between the EBLASSO and LASSO showed that the LASSO is slightly faster than the current version of EBLASSO, however, this may not be the case when the EBLASSO is implemented in C/C++ instead of Matlab. Although EB, LASSO, PENAL and SSVS methods all produced satisfactory results in a simulation study [12], the EB method outperformed the other three methods in terms of the mean-squared error. Moreover, when being applied to a real data set, the EB and LASSO detected some effects, whereas the PENAL and SSVS failed to generate any meaningful results [12]. In our simulation studies, we observed that the EBLASSO method detected more true effects than the EB method with almost the same false positive rate, and the same number of true effects as the LASSO but

with a much smaller number of false positive effects. When analyzing a real data set, we found that the EBLASSO method detected a reasonable number of effects, but the EB method detected one or zero effect depending on values of the hyperparameters used. These observations in both simulation study and real data analysis demonstrated that the EBLASSO method outperforms the EB method and the LASSO.

The EBLASSO method was built upon the idea of the RVM in machine learning. The EBLASSO and EB methods, as well as the RVM, all are based on a Bayesian hierarchical linear regression model and all estimate the variances of the regression coefficients. The difference of the three methods in the regression model is the different prior distributions for the hyperparameters. The EB method and the RVM employ inverse chi-square and uniform distributions, respectively, for the variances of the regression coefficients, while the EBLASSO assigns exponential distributions to the variance components and uses a Gamma distribution for the parameter of the exponential distribution, which leads to the prior distribution in (3) for the variance components. The uniform prior distribution used by the RVM may not provide enough degree of shrinkage for certain data and thus generate a large number of false positive effects as shown in [13] and as demonstrated in our simulation study.

The prior distributions used by the EBLASSO and RVM methods enable one to estimate the variance components in a closed form, while the EB method generally needs a numerical optimization algorithm to estimate the variance components. This difference has at least two implications: 1) both the EBLASSO and the RVM methods requires much less computation than the EB method to estimate the variance components, and 2) the EBLASSO method and the RVM method can always find the unique optimal estimate of a variance component but the numerical optimization algorithm used by the EB method may not find the optimal value of the variance due to the nonlinearity and non-convexity of the objective function. Another main factor that makes the EBLASSO method and RVM more efficient is an automatic variable selection procedure resulting from the process of estimating variance components, because the variables whose precision is infinity or equivalently whose variance is zero are excluded from the model. This results in an efficient formula in (12) for calculating the inverse of the covariance matrix of the data. This is especially beneficial when the number of samples is relatively large. On the other hand, the EB method in principle can be applied to a linear regression model with any prior distribution for the variances of regression coefficients. Since the prior distribution may play an important role in estimation of the QTL effects, the

EB method has its value when one tries to explore different prior distributions.

To get the best performance, the EBLASSO method needs to properly choose values of hyperparameters a and b. In this paper, we selected the values of a and b that gave the smallest average PE resulting from tenfold cross validation. Ideally, we need to find PEs for a large set of values for $a \geq -1$ and $b > 0$ and then identify the best values for a and b. In our simulation study, we found a two-step cross validation procedure could significantly reduce the number of values to be tested without missing the best values, thereby reducing computational time. In this two-step procedure, we first run cross-validation for the following set of values: $a = b = 0.001, 0.01, 0.05, 0.1, 0.5, 1$. We identify the values (denoted as a^* and b^*) from this set of values that yields the smallest PE. We then fixed b to be b^* and run cross-validation for several other values of a greater or less than a^*. The final best values of a and b are the ones that yield the smallest PE.

The EBLASSO algorithm may still be improved. In the analysis of simulated data with both main and epistatic effects, although the EBLASSO method detected 8 more true effects than the EB method without any false positive effects, it missed three effects that the EB method detected. It is unclear how this discrepancy occurred. One possible reason is the different prior distributions used in the two methods. Although it is difficult for the EBLASSO method to use the scaled inverse chi-square distribution that is used by the EB method, other prior distributions may worth investigation. Another possible reason may be the greedy method used to select the variable to include in or to exclude from the model. In the current algorithm, we choose the variable that gives the largest increase in the likelihood to add to or delete from the model. It may be better to simultaneously add or delete more than one variables. The EBLASSO method presented in this paper assumes continuous quantitative traits. It can also be extended to handle binary or polychotomous traits and the algorithm is under development. The algorithm is currently implemented in Matlab. We are developing programs in C++ to implement the algorithm, which is expected to be much faster and to be capable of running in R and SAS environments.

Conclusions

We have developed a fast empirical Bayesian LASSO method for multiple QTL mapping that can deal with a large number of effects possibly including main and epistatic QTL effects, environmental effects and the effects of environment and gene interactions. Our simulation studies demonstrated that the EBLASSO algorithm needed about 3.4 minutes of CPU time, running in Matlab on a PC with 2.4 GHz Intel Core2 CPU and 2 Gb memory running Windows XP, to analyze a QTL model with more than 10^5 possible effects, whereas the EB method took more than 2,000 minutes to analyze the same model on the same computer. Our simulation studies also showed that the EBLASSO method could detect more true effects with almost the same false positive rate comparing to the EB method. Our real data analysis demonstrated that the EBLASSO method could output more reasonable effects than the EB method. Comparing with the LASSO, our simulation showed that the current version of the EBLASSO implemented in Matlab was slightly slower than the LASSO implemented with glmnet in Fortran, and that the EBLASSO detected the same number of true effects as the LASSO but a much smaller number of false positive effects. In conclusion, the EBLASSO method will be a useful tool in multiple QTL mapping.

Author details

[1]Department of Electrical and Computer Engineering, University of Miami, Coral Gables, FL 33146, USA. [2]Department of Botany and Plant Sciences, University of California, Riverside, CA 92521, USA.

Authors' contributions

XC conceived the idea of the EBLASSO method, developed the EBLASSO algorithm, participated in the data analysis and drafted the manuscript. AH developed the computer programs and performed the data analysis. SX participated in the development of the EBLASSO algorithm, designed simulation study, participated in the data analysis and helped to draft the manuscript. All authors read and approved the final manuscript.

References

1. Carlborg Ö, Haley CS: **Epistasis: too often neglected in complex trait studies?** *Nat Rev Genet* 2004, **5**:618-625.
2. Schwarz G: **Estimating the dimension of a model.** *Ann Stat* 1978, **6**:461-464.
3. Hoerl AE, Kennard RW: **Ridge regression: Biased estimation for nonorthogonal problems.** *Technometrics* 1970, **12**:55-67.
4. Tibshirani R: **Regression Shrinkage and Selection Via the Lasso.** *J R Statistical Soc Ser B* 1996, **58**:267-288.
5. O'Hara RB, Sillanpää MJ: **A review of Bayesain variable selection methods: what, how and which.** *Bayesian Analysis* 2009, **4**:85-118.
6. Xu S: **Estimating polygenic effects using markers of the entire genome.** *Genetics* 2003, **163(2)**:789-801.
7. Wang H, Zhang YM, Li X, Masinde GL, Mohan S, Baylink DJ, Xu S: **Bayesian shrinkage estimation of quantitative trait loci parameters.** *Genetics* 2005, **170**:465-480.
8. Hoti F, Sillanpää MJ: **Bayesian mapping of genotype × expression interactions in quantitative and qualitative traits.** *Heredity* 2006, **97**:4-18.
9. Huang H, Eversley CD, Threadgill DW, Zou F: **Bayesian Multiple Quantitative Trait Loci Mapping for Complex Traits Using Markers of the Entire Genome.** *Genetics* 2007, **176(4)**:2529-2540.

10. Yi N, Xu S: **Bayesian LASSO for quantitative trait loci mapping.** *Genetics* 2008, **179(2)**:1045-1055.
11. Robert CR, Casella G: *Monte Carlo Statistical Methods.* 2 edition. Springer; 2004.
12. Xu S: **An Empirical Bayes Method for Estimating Epistatic Effects of Quantitative Trait Loci.** *Biometrics* 2007, **63(2)**:513-521.
13. Xu S: **An expectation maximization algorithm for the Lasso estimation of quantitative trait locus effects.** *Heredity* 2010, **2010**:1-12.
14. Nelder JA, Mead R: **A Simplex Method for Function Minimization.** *Comput J* 1965, **7**:308-313.
15. Tipping ME: **Sparse Bayesian learning and the relevance vector machine.** *J Mach Learn Res* 2001, **1**:211-244.
16. Tipping ME, Faul AC: **Fast Marginal Likelihood Maximisation for Sparse Bayesian Models.** *Proc 9th International Workshop on Artificial Intelligence and Statistics* Key West, FL; 2003.
17. Park T, Casella G: **The Bayesian Lasso.** *J Amer Stat Assoc* 2008, **103(482)**:681-686.
18. Yi N, Banerjee S: **Hierachical Generalized Linear Models for Multiple Quantitative Trait Locus Mapping.** *Genetics* 2009, **181**:1101-1133.
19. Friedman J, Hastie T, Tibshirani R: **Regularization Paths for Generalized Linear Models via Coordinate Descent.** *J Stat Softw* 2010, **33**:1-22.
20. Luo ZW, Potokina E, Druka A, Wise R, Waugh R, Kearsey MJ: **SEP genotyping from Affymetrix arrays is robust but largely detects cis-acting expression regulators.** *Genetics* 2007, **176**:789-800.
21. Zhang YM, Xu S: **A penalized maximum likelihood method for estimating epistatic effects of QTL.** *Heredity* 2005, **95**:96-104.
22. George EI, McCulloch RE: **Variable Selection Via Gibbs Sampling.** *Journal of the American Statistical Association* 1993, **88(423)**:881-889.
23. Yi N, George V, Allison DB: **Stochastic Search Variable Selection for Identifying Multiple Quantitative Trait Loci.** *Genetics* 2003, **164**:1129-1138.