# A Fast Algorithm for Maximum Likelihood Estimation of Mixture Proportions Using Sequential Quadratic Programming*

Youngseok Kim, Peter Carbonetto, Matthew Stephens

Department of Statistics, University of Chicago

and

Mihai Anitescu †

Department of Statistics, University of Chicago, and

Mathematics and Computer Science Division,

Argonne National Laboratory

June 8, 2018

**Abstract**

Maximum likelihood estimation of mixture proportions has a long history in statistics and continues to play a role in modern statistics, notably in recently developed non-parametric empirical Bayes (EM) methods. Although this problem has traditionally been solved by using an EM algorithm, recent work by Koenker and Mizera shows that

*Also, preprint ANL/MCS-P9073-0618, Argonne National Laboratory. This material was based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research (ASCR) under Contract DE-AC02-06CH11347. We acknowledge partial NSF funding through awards FP061151-01-PR and CNS-1545046 to MA, and support from NIH grant HG002585 and a grant from the Gordon and Betty Moore Foundation to MS. We also thank the University of Chicago Research Computing Center for providing high-performance computing resources used to implement some of the numerical experiments. Also, thanks to Joe Marcus for his help with processing the GIANT data set.

†corresponding author, anitescu@anl.gov

modern convex optimization methods can be substantially faster and more accurate. In particular, they used an interior point (IP) method to solve a dual formulation of the original optimization problem. Here we develop a new optimization approach, based on sequential quadratic programming (SQP), which is substantially faster than the IP method without sacrificing accuracy. Our approach combines several ideas: first, it solves a reformulation of the original primal problem rather than the dual; second, it uses SQP instead of IP, with the goal of making full use of the expensive gradient and Hessian computations; third, it uses an active set method to solve the QP subproblem within the SQP algorithm to exploit the sparse nature of the problem; and fourth, it uses low-rank approximations for much faster gradient and Hessian computations at virtually no cost to accuracy. We illustrate all these ideas in experiments on synthetic data sets as well as on a large genetic association data set. For large data sets (e.g., $n \approx 10^6$ observations and $m \approx 10^3$ mixture components) our implementation yields at least 100-fold reduction in compute time compared with state-of-the-art IP methods, and it does so without sacrificing accuracy. Our methods are implemented in Julia and R, and we have made the source code available at https://github.com/stephenslab/mixsqp-paper.

*Keywords:* nonparametric empirical Bayes, nonparametric maximum likelihood, mixture models, convex optimization, sequential quadratic programming, active set methods, rank-revealing QR decomposition

# 1 Introduction

We consider maximum likelihood estimation of the mixture proportions in a finite mixture model where the component densities are known. The simplest example of this arises if we have independent and identically distributed (*i.i.d.*) observations $z_1, \ldots, z_n$ from a finite mixture distribution with density

$$p(\cdot \mid x) = \sum_{k=1}^{m} x_k g_k(\cdot), \tag{1}$$

where the component densities $g_k(\cdot)$ are known and $x = (x_1, \ldots, x_m)^T$ denotes the unknown mixture proportions (non-negative, and summing to one). Finding the maximum likelihood estimate (MLE) of $x$ can be written as

$$\begin{aligned} \text{minimize} \quad & f(x) \triangleq -\frac{1}{n} \sum_{j=1}^{n} \log \left( \sum_{k=1}^{m} L_{jk} x_k \right) \\ \text{subject to} \quad & x \in \mathcal{S}^m \triangleq \left\{ x : \textstyle\sum_{k=1}^{m} x_k = 1,\ x \succeq 0 \right\}, \end{aligned} \tag{2}$$

where $L_{jk} \triangleq g_k(z_j) \geq 0$. The same optimization problem (2) arises in many other settings—including in nonparametric empirical Bayes (EB) analyses described later—where observations are not necessarily identically distributed. Here, we develop general methods for solving (2).

Problem (2) is a convex optimization problem and can be solved simply by using the expectation maximization (EM) algorithm (Dempster et al. 1977). However, the convergence of the EM algorithm can be intolerably slow in many mixture optimization problems (Redner & Walker 1984, Atkinson 1992, Salakhutdinov et al. 2003, Varadhan & Roland 2008), and this slow convergence is illustrated particularly evocatively in Koenker & Mizera (2014). Koenker & Mizera (2014) and Koenker & Gu (2017) point out that modern convex optimization methods can be substantially faster and more reliable than EM. They demonstrate this feature by using an interior (IP) method to solve a dual formulation of the original problem. This method is implemented in the `KWDual` function of the R package `REBayes` (Koenker & Gu 2017), which interfaces to the commercial interior point solver `MOSEK` (Andersen & Andersen 2000).

In this paper, we provide an even faster algorithm for this problem based on sequential quadratic programming (SQP) (Nocedal & Wright 2006), and we implement it entirely in an open source setting using the Julia programming language (Bezanson et al. 2012). The computational gains are greatest for large data sets, in settings where the likelihood matrix $L \in \mathbb{R}^{n \times m}$ is numerically rank deficient. Rank deficiency can make the optimization problem harder to solve in practice, even if it is convex (Wright 1998). As we show later, a numerically rank-deficient $L$ often occurs in the nonparametric EB problems that are the primary focus of `KWDual`. As an example of target problem size, we later consider data from a genome-wide association study with $n > 10^6$ and $m > 100$. For such data, our methods are at least 100 times faster than `KWDual` (about 10 s vs. $10^3$ s). Code implementing our methods and used in numerical experiments is available in the supplementary materials and online at https://github.com/stephenslab/mixsqp-paper.

# 2 Motivation: Nonparametric empirical Bayes problems

Estimation of mixture proportions is a fundamental problem in statistics, dating back to at least Pearson (1894). This consideration, combined with the need to fit increasingly large data sets, already provides strong motivation for finding efficient scalable algorithms for this problem. Here, however, we are particularly motivated by recent work on nonparametric approaches to EB estimation (Koenker & Mizera 2014, Stephens 2016) where finite mixtures with a large number of components are used to accurately approximate *nonparametric* families of prior distributions. Here we briefly discuss this motivating application.

We first consider a simple EB problem based on the "normal means" or "Gaussian sequence" problem (Johnstone & Silverman 2004). For $j = 1, \ldots, n$, we observe data $z_j$ that are noisy observations of some underlying "true" values $\theta_j$, with normally distributed errors of known variance $s_j^2$:

$$z_j \,|\, \theta_j \sim N(\theta_j, s_j^2). \tag{3}$$

The EB approach to this problem assumes that $\theta_j$ are *i.i.d.* from some unknown distribution

$g$,

$$\theta_j \,|\, g \sim g, \quad g \in \mathcal{G}, \tag{4}$$

where $\mathcal{G}$ is some pre specified class of distributions. The EB approach estimates $g$ by maximizing the (marginal) log-likelihood:

$$\underset{g \in \mathcal{G}}{\text{minimize}} \; -\frac{1}{n} \sum_{j=1}^{n} \int \mathcal{N}(z_j; \theta, s_j^2)\, g(\theta)\, d\theta. \tag{5}$$

where $\mathcal{N}(\cdot; \mu, \sigma^2)$ denotes the normal density with mean $\mu$ and variance $\sigma^2$. It then computes the posterior distribution for each $\theta_j$ given the estimated $g$. Our focus here is on this maximization step.

Although one can use simple parametric families for $\mathcal{G}$, in many settings one might prefer to use a more flexible nonparametric family. Examples include the following:

- $\mathcal{G} = \mathcal{R}$, the set of all real-valued distributions.
- $\mathcal{G} = \mathcal{U}_0$, the set of unimodal distributions with a mode at zero. (Extensions to nonzero mode are straightforward.)
- $\mathcal{G} = \mathcal{SU}_0$, the set of symmetric unimodal distributions with a mode at 0.
- $\mathcal{G} = \mathcal{SN}_0$, the set of distributions that are scale mixtures of zero-mean normals (which includes several commonly used distributions, including $t$, and double-exponential or Laplace.)

The fully nonparametric case $\mathcal{G} = \mathcal{R}$ is well studied (e.g., Laird 1978, Jiang & Zhang 2009, Brown & Greenshtein 2009, Koenker & Mizera 2014), and is related to the classic Kiefer-Wolfowitz problem (Kiefer & Wolfowitz 1956). The more constrained cases $\mathcal{G} = \mathcal{U}_0, \mathcal{SU}_0, \mathcal{SN}_0$ appear in Stephens (2016) (see also Cordy & Thomas 1997) and can be motivated by the desire to shrink estimates towards zero, as well as a desire to impose some regularity on $g$ without making strong parametric assumptions.

The connection with (2) is that these nonparametric sets can be approximated, arbitrarily accurately, by using a finite mixture with sufficiently large number of components.

That is, they can be approximated by

$$\mathcal{G} \triangleq \left\{ g = \sum_{k=1}^{m} x_k g_k : \sum_{k=1}^{m} x_k = 1,\ x \succeq 0 \right\}, \tag{6}$$

for some choice of distributions $g_k$, $k = 1, 2, \ldots, m$. The $g_k$'s are often called *dictionary functions* (Aharon et al. 2006). For example,

- $\mathcal{G} = \mathcal{R}$: $g_k = \delta_{\mu_k}$ where $\delta_\mu$ denotes a delta-Dirac point mass at $\mu$, and $\mu_1, \ldots, \mu_m \in \mathbb{R}$ denotes a suitably fine grid of values across the real line.
- $\mathcal{G} = \mathcal{U}_0$, $\mathcal{G} = \mathcal{SU}_0$: $g_k = \text{Unif}[0, a_k], \text{Unif}[-a_k, 0]$ or $\text{Unif}[-a_k, a_k]$, where $a_1, \ldots, a_m \in \mathbb{R}^+$ is a suitably fine grid of values.
- $\mathcal{G} = \mathcal{SN}_0$: $g_k = N(0, \sigma_k^2)$, where $\sigma_1^2, \ldots, \sigma_m^2 \in \mathbb{R}^+$ is a suitably fine grid of values.

With these approximations, fitting (3) then boils down to solving an optimization problem of the form (2) with $L_{jk} = \int \varphi(z_j; \theta, s_j^2)\, g_k(d\theta)$.

A key feature of these problems is that they all use a fine grid to approximate a nonparametric family. The result is that *many of the distributions $g_k$ are similar to one another.* Hence, the matrix $L$ is numerically rank deficient; and, in our experience, many of its singular values are near floating-point machine precision. We pay particular attention to this feature when designing our optimization methods.

The normal means problem is just one example of a broader class of problems with similar features. The general point is that nonparametric problems can often be accurately solved with finite mixtures, resulting in optimization problems of the form (2), typically with moderately large $m$, larger $n$, and a numerically rank-deficient $n \times m$ matrix $L$.

# 3 A new approach based on SQP

The methods from Koenker & Gu (2017) (implemented by function `KWDual` in R package `REBayes`), provide, to our knowledge, the best current implementation for solving (2).

These methods are based on reformulating (2) as

$$\text{minimize} \quad -\frac{1}{n}\sum_{j=1}^{n}\log y_j \quad \text{subject to} \quad Lx = y, \quad x \in \mathcal{S}^m, \tag{7}$$

then solving the dual ("K-W dual" in Koenker & Mizera (2014)),

$$\text{minimize} \quad -\frac{1}{n}\sum_{j=1}^{n}\log \nu_j \quad \text{subject to} \quad L^T\nu \preceq n\mathbb{1}_m, \quad \nu \in R^n \succeq 0, \tag{8}$$

where $\mathbb{1}_m$ is the vector of ones of dimension $m$. Koenker & Mizera (2014) report that solving the dual formulation was generally faster than primal formulations in their assessments. Indeed, we also found this to be the case for IP-based approaches (see Figure 1). For $n \gg m$, however, we believed that the original formulation (2) offered more potential for improvements. In particular, in the dual formulation (8), efforts depend on $n$ when computing the gradient and Hessian of the objective (admittedly, a diagonal matrix), when evaluating the constraints, and when computing the Newton step inside the interior point algorithm. By contrast, in (2) efforts depend on $n$ only in the gradient and Hessian computations. All other evaluations depend on $m$ only.

These considerations motivated the design of our algorithm, which was developed with two key principles in mind: (i) make the best possible use of each expensive Hessian computation in order to minimize the number of Hessian evaluations; and (ii) reduce the expense of each Hessian evaluation as much as possible. (We could have avoided Hessian computations entirely by pursuing a first-order optimization method, but we judged that a second-order method would likely be more robust and stable because of the ill-conditioning caused by the numerical rank deficiency of $L$.)

To make the best use of each Hessian computation, we apply sequential quadratic programming (Nocedal & Wright 2006) to a reformulation of the primal problem (2). SQP attempts to make best use of the (expensive) Hessian computations by finding, at each iteration, the best reduction in a quadratic approximation to the constrained optimization problem. Furthermore, our reformulation relaxes the simplex constraint to a less restrictive non-negative one, which simplifies computations as well.

To reduce the computational cost of each Hessian evaluation, we exploit the numerical low-rank of $L$, using a "rank-revealing" QR (RRQR) decomposition of $L$ (Golub & Van Loan 2012). This matrix decomposition, which need only be performed once, reduces subsequent per iteration computations so that they depend on the numerical rank $r$ rather than $m$. Specifically, Hessian computations are reduced from $O(nm^2)$ to $O(nr^2)$.

In addition to these two key principles, we paid some attention to optimizing certain details. In particular, based on initial findings that the primal solution is typically sparse, we implemented an active set method—one that estimates which entries of the solution are zero—to solve for the search direction at each iteration of the SQP. As we show later, an active set approach effectively exploits the solution's sparsity.

The remaining subsections detail each of these innovations in turn.

## 3.1 A reformulation

We reformulate problem (2) into a simpler problem with less restrictive non-negative constraints using the following definition and theorem.

**Definition 3.1.** *A function $\phi : \mathbb{R}^m_+ \to \mathbb{R}$ is said to be "scale invariant" if for any $c > 0$ there exists a $C \in \mathbb{R}$ such that for any $x \in \mathbb{R}^m_+$ we have $\phi(cx) = \phi(x) - C$.*

**Theorem 3.2.** *Consider the simplex-constrained optimization problem*

$$\text{minimize} \quad \phi(x) \quad \text{subject to} \quad x \in \mathcal{S}^m, \tag{9}$$

*where $\phi(x)$ is scale invariant, convex, and nonincreasing with respect to $x$—that is, $x \succeq y$ (the componentwise partial ordering) implies $\phi(x) \leq \phi(y)$ for all $x \in \mathbb{R}^m_+$. Let $x^*(\lambda)$ denote the solution to the (partial) Lagrangian relaxation of problem (9),*

$$\text{minimize} \quad \phi_\lambda(x) \triangleq \phi(x) + \lambda \sum_{k=1}^{m} x_k \quad \text{subject to} \quad x \in \mathbb{R}^m_+ \triangleq \{x \in \mathbb{R}^m : x \succeq 0\} \tag{10}$$

*Then $x^* \triangleq x^*(\lambda) \Big/ \sum_{k=1}^m x_k^*(\lambda)$ is a solution to (9).*

Setting $\phi$ to be the objective function in (2) yields the following corollary.

**Corollary 3.3.** *The target optimization problem* (2) *can be solved by instead solving* (10) *with* $\phi(x) = f(x)$ *in* (2)*; that is:*

$$\text{minimize} \quad f^\star(x) \triangleq f(x) + \sum_{k=1}^{m} x_k \quad \text{subject to} \quad x \succeq 0. \tag{11}$$

*Furthermore, in this case setting* $\lambda = 1$ *yields a solution that is normalized; in other words,* $x^* = x^*(\lambda)$ *when* $\lambda = 1$.

The proof of both the theorem and the corollary are in the Appendix. Although here we focus specifically on the case $\phi = f$, many of our ideas should apply more generally for other $\phi$ satisfying the properties of Theorem 3.2. For example, consider the case when $f$ is a composite of "easily differentiable" scale-invariant function and "thin and tall" linear functions. The algorithmic ideas illustrated in Section 3 are still applicable to those functions. See the Appendix for further discussion.

## 3.2 Sequential quadratic programming

We solve (11) using an SQP algorithm with backtracking line search (Nocedal & Wright 2006). In brief, SQP is an iterative algorithm that, at the $t$-th iteration, formulates a second-order approximation of the objective at the feasible point $x^{(t)}$, then determines a search direction $p^{(t)}$ based on the second-order approximation. Specifically, the search direction $p^{(t)}$ is obtained by solving the following non-negatively-constrained quadratic program, henceforth called the "QP subproblem":

$$p^{(t)} = \arg\min_p \tfrac{1}{2} p^T H_t p + p^T g_t \quad \text{subject to} \quad x^{(t)} + p \succeq 0, \tag{12}$$

where $g_t = \nabla f^\star(x^{(t)})$ and $H_t = \nabla^2 f^\star(x^{(t)})$ are the gradient and Hessian of $f^\star(x)$ at $x^{(t)}$. Computation of the gradient and Hessian is considered in Section 3.3; solving the QP subproblem is described in Section 3.4.

After identifying the search direction $p^{(t)}$, we perform a backtracking line search to determine a sufficient descent step $x^{(t+1)} = x^{(t)} + \alpha_t p^{(t)}$ for $\alpha_t \in [0, 1]$. In contrast to other projection-based methods such as the projected Newton method (Kim et al. 2010),

$x^{(t+1)}$ is guaranteed to be (primal) feasible for all choices of $\alpha_t \in [0,1]$ provided $x^{(t)}$ is feasible. This is due to the linearity of the inequality constraints. As discussed in Nocedal & Wright (2006), the line search will accept unitary steps ($\alpha_t = 1$) close to the solution, and the algorithm will be quadratically convergent provided the reduced Hessian is positive definite at the optimal active set (*i.e.,* for the indices that are nonzero at the solution $x^*$). A similar argument can be found in Wright (1998).

### 3.3 Gradient and Hessian evaluations

We now discuss computation of the gradient and Hessian and ways to reduce their computational burden.

**Lemma 3.4.** *For all $x \in \mathbb{R}^m_+$, the gradient and Hessian of the objective function in (11) at $x$ are*

$$g = -\tfrac{1}{n} L^T d + \mathbb{1}_m, \quad H = \tfrac{1}{n} L^T \mathrm{diag}(d)^2 L, \tag{13}$$

*where $\mathbb{1}_m$ is the vector of ones of dimension $m$ and where $d = (d_1, \ldots, d_n)^T$ is the column vector with entries $d_j = 1/(Lx)_j$. Furthermore, for all $x \in \mathbb{R}^m_+$, we have that*

$$x^T g = 0, \quad x^T H x = 1, \quad Hx + g = \mathbb{1}_m. \tag{14}$$

The proof of the Lemma is in the Appendix. Since $L$ is, in general, a dense $n \times m$ matrix, computing $d$ requires $O(nm)$ multiplications, as does computing $g$. Computing $H$ is more expensive, requiring $O(nm^2)$ multiplications, as discussed above.

In practice, we find that $L$ is often numerically rank deficient, with (numerical) rank $r < m$. We can exploit this to reduce computational complexity by approximating $L$ by a low-rank matrix. Here we use either the RRQR decomposition (Golub & Van Loan 2012) or truncated singular value decomposition (tSVD) to compute highly accurate low-rank approximations to $L$. Specifically, we use the `pqrfact` and `psvdfact` functions from the `LowRankApprox` Julia package (Ho & Olver 2018), which implement randomized algorithms based on Halko et al. (2011).

Given the RRQR approximation of $L$, $\tilde{L} = QRP^T$, with $Q \in \mathbb{R}^{n\times r}, R \in \mathbb{R}^{r\times m}$ and

permutation matrix $P \in \mathbb{R}^{m\times m}$, the gradient and Hessian can be approximated as

$$\tilde{d}_j = 1/(QRP^T x)_j, \quad \tilde{g} = -\tfrac{1}{n} PR^T Q^T d + \mathbb{1}_m, \quad \tilde{H} = \tfrac{1}{n} PR^T Q^T \mathrm{diag}(d)^2 QRP^T. \tag{15}$$

Corresponding expressions for the tSVD are straightforward to obtain, and we therefore omit them here.

Once we have obtained a QR (or SVD) approximation, the key to reducing the expense of the gradient and Hessian computations is to avoid directly reconstructing $\tilde{L}$. For example, the computation of the gradient $\tilde{g}$ is implemented as $-(((d^T Q)R)P^T)^T/n + \mathbb{1}$ so that all matrix operations are a matrix times a vector. The dominant cost in (15) is computation of the product $Q^T \mathrm{diag}(d)^2 Q$, which needs $O(nr^2)$ multiplications. Overall, computation is reduced by roughly a factor of $(r/m)^2$ per iteration compared with (13). To enjoy this benefit, we pay the one-time cost of factorizing $L$, which, in the regime $n \gg m$, is $O(nmr)$ (Golub & Van Loan 2012).

In replacing $L$ with $\tilde{L} = QRP^T$ some numerical approximation is inevitable, and we are effectively solving an approximation to (11),

$$\text{minimize} \quad \tilde{f}(x) \triangleq -\frac{1}{n}\sum_{j=1}^{n} \log\left(\sum_{k=1}^{m} \tilde{L}_{jk} x_k\right) + \sum_{k=1}^{m} x_k \quad \text{subject to} \quad x \succeq 0. \tag{16}$$

Because of the finite precision of the low-rank approximation, the quantity $\sum_{k=1}^{m} \tilde{L}_{jk} x_k$ can be slightly below zero. This can cause numerical issues since the logarithm in the objective does not accept negative values, for example when the objective of (16) is evaluated during line search. To avoid these numerical issues, we add a small positive constant (e.g., $10^{-8}$) to the argument of the logarithm when evaluating $\tilde{f}(x)$.

### 3.4 Solving the QP subproblem

To solve (12), we solve the equivalent problem

$$y^* = \arg\min_y \tfrac{1}{2} y^T H_t y + y^T a_t \quad \text{subject to} \quad y \succeq 0, \tag{17}$$

where $a_t = -H_t x^{(t)} + g_t$, which $= 2g_t - \mathbb{1}_m$ from (14). This problem is derived by substituting $y = x^{(t)} + p$ into (12). It is straightforward to show that setting $p^* = y^* - x^{(t)}$ solves (12). The simpler non-negativity constraints make (17) easier to solve.

To solve (17), we implement an active set method following (Nocedal & Wright 2006, §16.5). This method starts from a feasible point $y^{(0)}$ and $\mathcal{W}^{(0)}$, an initial guess of the active set at the solution (the "working set"). The initial $y^{(0)}$ can be set to a fixed value—for example, $(1, 0, \cdots, 0)^T$ with $\mathcal{W}^{(0)} = [m] \backslash \{1\}$—or it can be determined from the previous SQP outer-loop iteration $x^{(t)}$, with $\mathcal{W}^{(0)} = [m] \backslash \text{supp}(y^{(t)})$ (this is often called "warm-starting"). For all results in this paper we initialized the active set solver at $\mathbb{1}_m/m$ the first time it was applied, and we used a warm start in subsequent iterations.

The active set method is an iterative method in which the $l$th iteration involves identifying a search direction $q^{(l)}$ by solving the following equality-constrained subproblem:

$$q^{(l)} = \arg\min_q \tfrac{1}{2} q^T H_t q + q^T b_l \quad \text{subject to} \quad q_i = 0, \quad \forall i \in \mathcal{W}^{(l)}, \tag{18}$$

such that $b_l = H_t y^{(l)} + a_t$. Solving (18) involves simply solving a linear system corresponding to inactive indices outside of the working set. Therefore, if the number of inactive indices remains much smaller than $m$ over the iteration, we expect a total number of floating-point operations much smaller than $O(m^3)$.

Additional implementation details, including steps to update the working set in the presence of *blocking-constraints*, are given in the Appendix.

### 3.5 The MIX-SQP algorithm

Putting these components together results in Algorithm 1, which we call MIX-SQP. We implemented this algorithm in the Julia programming language. The pseudocode describes the algorithm assuming that RRQR is used to approximate $L$, but this approximation step can be omitted or can be easily replaced with a truncated SVD approximation.

**Algorithm 1:** MIX-SQP

**Inputs:** likelihood matrix $L \in \mathbb{R}^{n \times m}$, initial iterate $x^{(0)} \in \mathbb{R}^m_+$,
sufficient decrease parameter $0 < \xi < 1$ (default is 0.5),
step size reduction $0 < \rho < 1$ (default is 0.5),
convergence tolerance $\epsilon > 0$ (default is $10^{-8}$)

$Q, R, P \leftarrow$ `pqrfact`$(L)$ `/* RRQR factorization */`

**for** $t = 0, 1, 2, \ldots$ **do**

  $d_t \leftarrow 1./(QRP^T x^{(t)})$ `/* "./" means elementwise division */`

  $g_t \leftarrow \frac{1}{n} PR^T Q^T d_t + \mathbb{1}$ `/* compute approximate gradient */`

  $H_t \leftarrow \frac{1}{n} PR^T Q^T \mathrm{diag}(d_t)^2 QRP^T$ `/* compute approximate Hessian */`

  $p_t \leftarrow$ MIX-ACTIVE-SET$(x^{(t)}, g_t, H_t)$ `/* solve QP subproblem (Alg. 2) */`

  **if** $\min_k |(g_t)_k| < \epsilon$ **and** $\|p_t\| < \epsilon$ **then**

    **break** `/* iterate satisfies convergence criteria */`

  $\alpha_t \leftarrow 1$ `/* backtracking line search */`

  **while** $\tilde{f}(x^{(t)} + \alpha_t p_t) > \tilde{f}(x^{(t)}) + \alpha_t \xi p_t^T g_t$ **do**

    $\alpha_t \leftarrow \rho \alpha_t$

  $x^{(t+1)} \leftarrow x^{(t)} + \alpha_t p_t$

**return** MIX-SQP$(L, x_0) = x^{(t)}$

# 4 Numerical experiments

We conducted numerical experiments to compare different methods for solving problems of the form (2). Specifically, we considered problems of this form that arise from nonparametric empirical Bayes with $\mathcal{G} = \mathcal{SN}_0$ (Section 2). Our comparisons involve many synthetic datasets with varying numbers of data points ($n$) and grid sizes ($m$). We also evaluate the methods on one large real dataset.

For the synthetic data, we generated $z_1, \ldots, z_n$ independently from

$$z_j \,|\, \theta_j, s_j^2 \sim N(\theta_j, s_j^2 = 1), \tag{19}$$

where the means $\theta_j$ are *i.i.d.* random draws from $g$, a heavy-tailed symmetric distribution about zero,

$$g = 0.5\mathcal{N}(0, 1) + 0.2t_4 + 0.3t_6, \tag{20}$$

and $t_\nu$ denotes the $t$ density with $\nu$ degrees of freedom.

For the real data, we used data from a genetic study of human height (Wood et al. 2014) produced by the GIANT ("Genetic Investigation of ANthropometric Traits") consortium.

Specifically, we used the estimated additive effects $z_j$ of $n$ = 2,126,678 genetic variants (single nucleotide polymorphisms, or SNPs) on human height. The data consist of the $n$ estimated effect sizes and their corresponding standard errors. For illustration purposes we treat the $n$ data points as independent here, although in practice there are local correlations between SNPs. See the Appendix for more details on steps taken to download and prepare the GIANT data for our experiments.

Under the model used in all the experiments—normal likelihood and a finite mixture-of-normals prior ($\mathcal{G} = \mathcal{SN}_0$)—the entries of the likelihood matrix are $L_{jk} = N(0, \sigma_k^2 + s_j^2)$. For the simulated data, $s_j = 1$, and for the GIANT data the $s_j$'s were set to the provided standard errors. The grid of variances $\sigma_1^2, \ldots, \sigma_m^2$ was adapted to each data set by using the default method from the `ashr` package (Stephens et al. 2018). In order to avoid overflow or underflow, each row of $L$ was computed up to a constant of proportionality such that the largest entry was always 1.

## 4.1 Approaches considered

The approaches evaluated in our experiments are combinations of the following elements:

- *Problem formulation:* The method solves the dual problem (8) (this is the problem solved by `KWDual`), the simplex-constrained problem (2), or the non-negatively-constrained optimization problem (11). This choice is indicated by `D`, `S`, or `NN`, respectively.

- *Solver:* The problem is solved by using either an SQP algorithm, Section 3.2, or an IP solver. This choice is denoted by `SQP` and `IP`, respectively.

- *QP Solver:* For the SQP method only, we consider two methods for solving the QP subproblem (12): the active set method described in Section 3.4 or an off-the-shelf QP solver (the commercial IP solver `MOSEK`). We indicate this choice with `A` or `IP`, respectively. When the SQP method is not used, we indicate this choice with `NA`, for "not applicable."

- *Gradient/Hessian computation:* The objective and partial derivatives either are computed exactly by using the full matrix $L$ (that is, within the accuracy limits of floating-

point operations), or they are approximated by using SVD or RRQR decompositions of $L$ (Section 3.3). We denote this choice by `F` (for the "full" matrix), `SVD` or `QR`.

A complete specification of a mixture proportions solver is denoted as

`[Formulation]-[Solver]-[QP Solver]-[Gradient/Hessian computation]`.

For example, the proposed MIX-SQP algorithm with a QR low-rank approximation to $L$ is written as `NN-SQP-A-QR`.

All experiments were run in the Julia interactive environment (version 0.6.2) on a machine with an Intel Xeon ("Broadwell") Processor E5-2680v4. The `KWDual` function in R package `REBayes` was called in R 3.4.1 and was interfaced to Julia using the `RCall` Julia package. Note that, for comparisons on the largest matrices ($n > 10^6$, $m = 800$), 16 GB of memory or more may be needed to load these matrices into Julia or R. Source code implementing all the methods compared in our experiments, including Jupyter notebooks illustrating how to use them, is available at https://github.com/stephenslab/mixsqp-paper.

## 4.2 Results

### 4.2.1 Comparing problem formulations

First, we investigated the benefits of the three problem formulations: the dual form (8), the simplex-constrained form (2), and the non-negatively-constrained form (11). In the `JuMP` modeling environment (Dunning et al. 2017), we implemented an IP approach and an SQP approach for each of the problem formulations and applied the $2 \times 3 = 6$ solvers to simulated data sets with $m = 40$ and with $n$ ranging from 80 to 40,960. For all SQP methods, an IP algorithm was also used to solve the QP subproblems; in short, this experiment compared solvers $x$`-IP-NA-F` and $x$`-SQP-IP-F`, with $x$ either `D`, `S` or `NN`. In all cases, the commercial solver `MOSEK` was used to implement the IP method. To provide a benchmark for comparison, when solving the dual problem, we also ran the `KWDual` method in R, which also calls `MOSEK`. `KWDual`/`MOSEK` was called from Julia, which generated the matrix $L$ for all experiments.

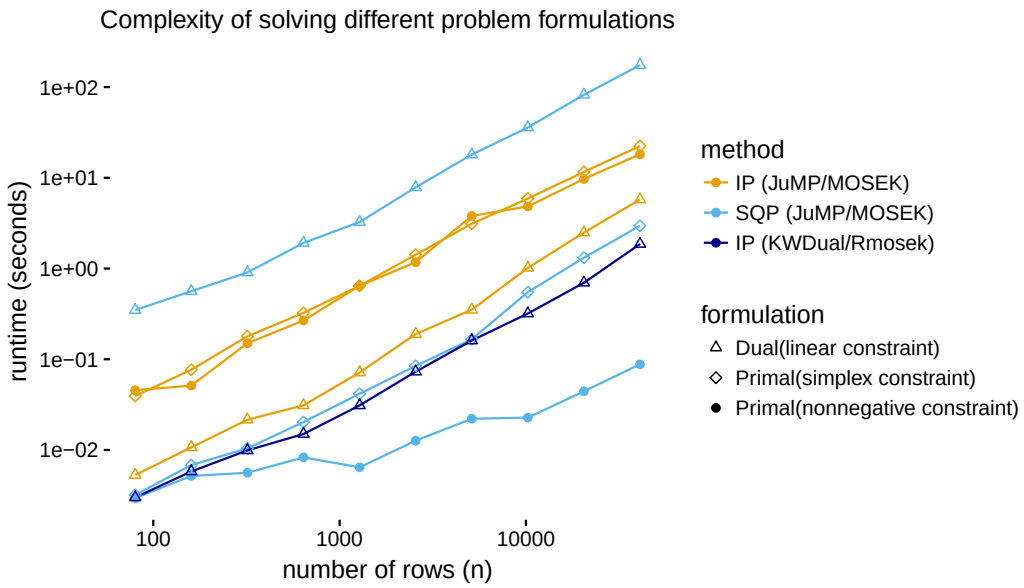


Figure 1: Runtimes for different formulations of the maximum-likelihood estimation problem: dual (8), simplex-constrained (2), and non-negatively-constrained (11). For each problem formulation, we applied an IP or SQP-based algorithm. We compared these with the `KWDual` function from the `REBayes` package, which solves the dual formulation using an R interface to `MOSEK`. Results are shown for $m = 40$ with varying $n$. Each timing is an average over 10 independently simulated data sets.

The results of this first experiment are shown in Figure 1. All methods had runtimes that scaled approximately linearly in $n$ (a slope on the log-log scale near 1). However, the SQP applied to the non-negatively-constrained reformulation (Section 3.1), `NN-SQP-IP-F`, was a clear winner. Further, it was substantially faster than SQP for the simplex-constrained problem. This is attributed in part to the non-negatively-constrained version requiring fewer outer iterations, although we cannot currently fully account for this improved convergence. Figure 1 also recapitulates the result from Koenker & Mizera (2014) that the IP method is faster when applied to the dual formulation.

Based on these results, we focused on the non-negatively-constrained formulation in our remaining experiments.

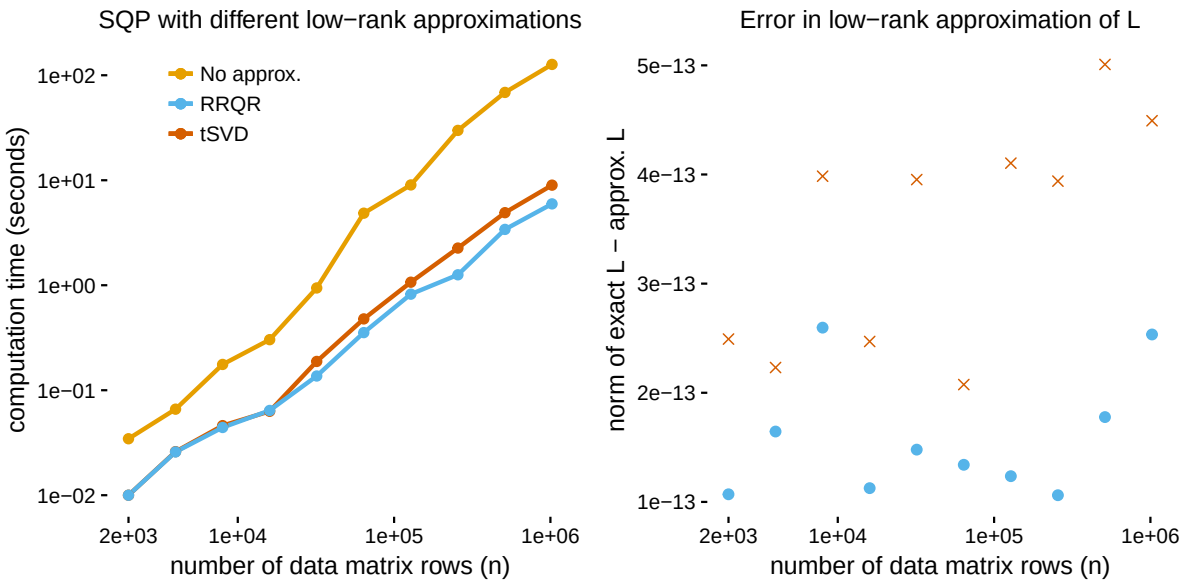


Figure 2: Comparison of SQP methods with and without exploiting the numerically low-rank structure of the $n \times m$ matrix $L$. *Left panel:* Runtime of the SQP solver using the full matrix $L$ against the same solver using low-rank approximations based on RRQR and tSVD factorizations of $L$. Here, $m = 100$. *Right panel:* Error in RRQR (circles) and tSVD ($\times$'s) reconstructions, $\tilde{L}$, in the same simulated data sets. Error is measured as $\|\tilde{L} - L\|_F$. Timings and errors are averages over 10 simulations.

### 4.2.2 Gains from low-rank approximations to $L$

Next, we investigated the benefits of exploiting numerically low-rank structure of $L$ (discussed in Section 3.3). Specifically, we compared runtime of the SQP method with and without low-rank (RRQR, tSVD) approximations, with varying numbers of samples, $n$; that is, we compared `NN-SQP-A-`$x$ solvers, with $x$ being one of `F`, `QR`, or `SVD`. To compute the RRQR and tSVD factorizations of $L$, we used functions `pqrfact` and `psvdfact` with the same relative tolerance, $10^{-10}$.

The results in Figure 2 show that the SQP method using RRQR and tSVD approximations was consistently faster by a large margin—a factor of 10 at $n = 10^6$—than computation with the full matrix $L$. For the largest $n$, SQP with RRQR was slightly faster than with tSVD. This is attributed mainly to the faster computation of the RRQR factorization over tSVD. Further, the improvement in computation time came at little expense to solution accuracy (Figure 2, right-hand panel). We found that the SQP method followed approx-

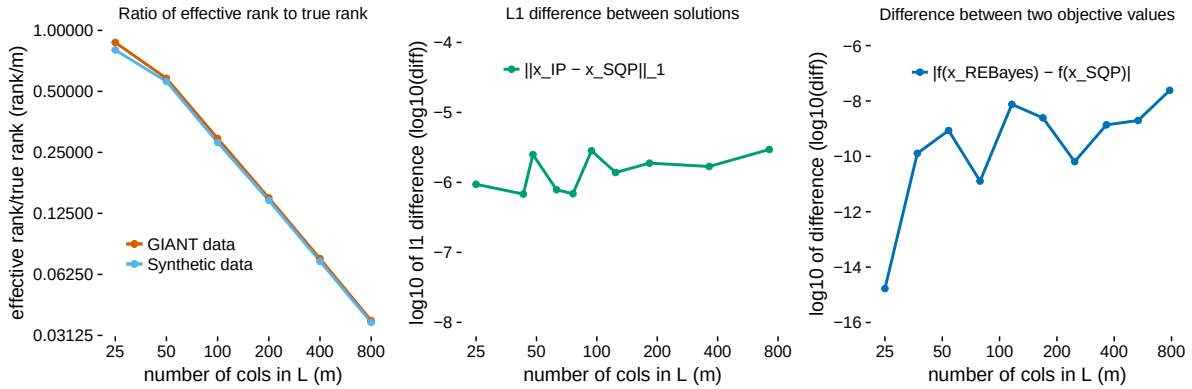


Figure 3: Assessment of numeric rank of $L$ and its effect on optimization performance. *Left panel:* The ratio of effective rank $r$ (numerical rank as estimated by RRQR) over $m$ for synthetic and GIANT data and data sets. *Middle panel:* $\ell_1$ norm of differences in solutions returned by `D-IP-NA-F` and `NN-SQP-A-QR` solvers applied to GIANT data set. *Right panel:* Difference in the objective values at these solutions.

imately the same path to the solution regardless of the low-rank approximation method used.

This speedup at little cost to accuracy was possible because the effective rank of $L$ ($r$) was small relative to $m$ in these simulated examples. To check that this was not an artifact of our data simulations, we also applied the same SQP method with RRQR-based derivative computations (`NN-SQP-A-QR`) to the GIANT data set. In both simulated and real data, the ratio $r/m$ was similar (Figure 3, left-hand panel).

We also assessed the impact of the low-rank approximations on the quality of the solution. For this experiment, we compared results on the GIANT data set. When using the RRQR approximation, the solution was close to what was obtained from the SQP method without approximation; the $\ell_1$ norm of the differences was on the order of $10^{-6}$, and the absolute difference in the objective values (log-likelihoods) was always less than $10^{-8}$ (Figure 3, middle and right-hand panels). Further, the RRQR approximation to $L$ led to equally sparse solutions (results not shown).

### 4.2.3 Comparing active set versus IP solutions to the QP subproblem

In this part, we compared different approaches to the QP subproblem step inside the SQP algorithm: an active set method (Section 3.4) and an off-the-shelf IP method (`MOSEK`).

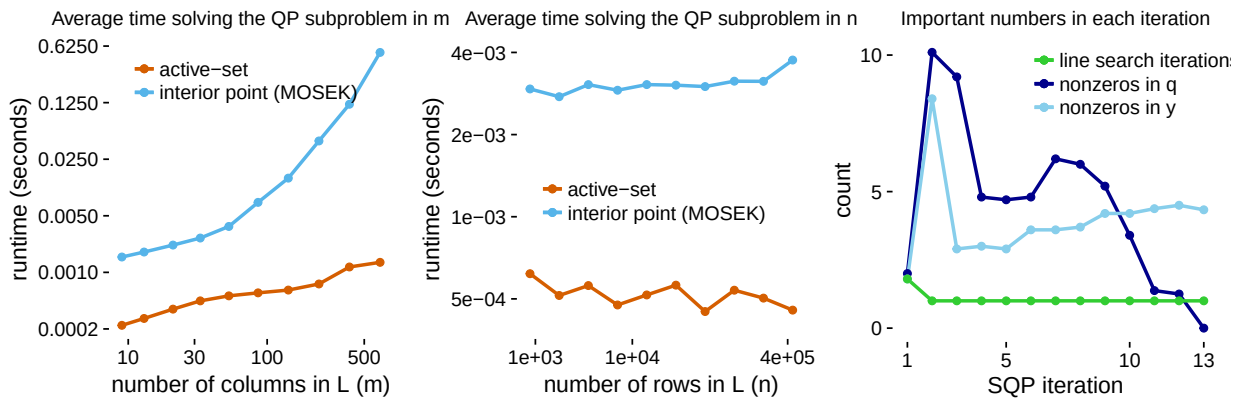


Figure 4: Comparison of active set and IP methods for solving QP subproblem inside SQP (17). *Left panel:* Runtimes of active set and IP (`MOSEK`) methods, with $n = 10^5$ and varying $m$. Timings were taken as averages over all subproblems solved until convergence of the SQP is reached, and over 10 simulations. *Middle panel:* Runtimes of IP and active set methods, with $m = 40$ and varying $n$. *Right panel:* Number of backtracking line searches (green), and the number of nonzeros in $q$ and $y$ on convergence of the active set method (dark and light blue), averaged over all SQP iterations, and over 10 data sets. See Section 3.4 for definitions of $q$ and $y$.

That is, we compared `NN-SQP-A-F` against `NN-SQP-IP-F`, recording only the time spent solving the QP subproblems. We set a stopping tolerance of $10^{-8}$ for both the IP and the active set approach (the latter uses a convergence criterion based on the first-order Karush-Kuhn-Tucker conditions).

The left and middle plots in Figure 4 show that the active set method consistently outperforms the IP method by a factor of roughly 5 or greater, with the greatest speedups achieved when both $m$ and $n$ are large. For example, when $n = 10^5$ and $m \approx 500$ (left-hand plot), the active set solver is over 100 times faster than the IP method.

We hypothesize that the active set is faster because the QP subproblem iterates and final solution are sparse and therefore the support of the variables usually includes only a small fraction of the co-ordinates (see right-hand plot in Figure 4 for an illustration); recall that the reformulated problem (11) and the QP subproblem (12) both have a non-negative constraint, which promotes sparsity. The left-hand plot in Figure 4 shows that the active set solution to the QP subproblem grows linearly with $m$, whereas the IP solution grows quadratically. By contrast, the average computation time of solving the QP subproblems

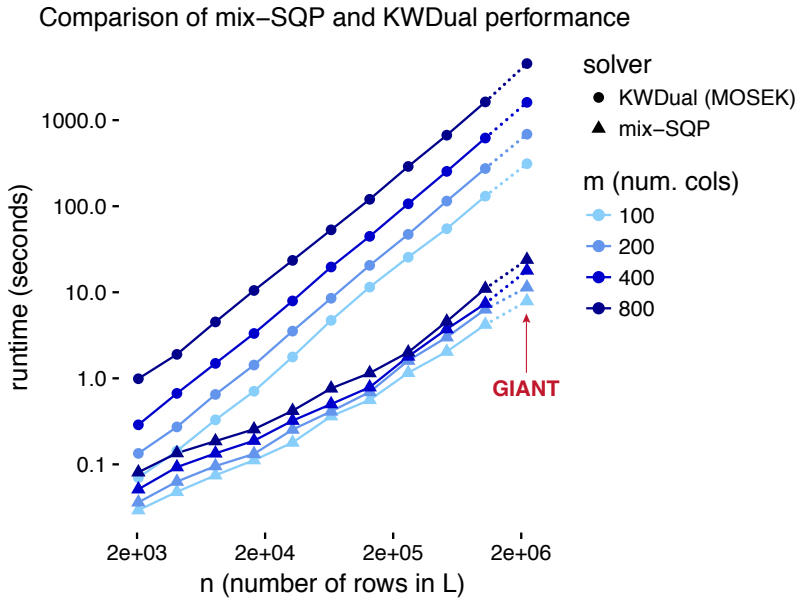


Figure 5: Runtimes of MIX-SQP and the `KWDual` (`MOSEK`) interior point solver applied to simulated data sets with varying $n$ and $m$, and to the GIANT data set ($n$ = 2,126,678). All runtimes on simulated data sets were taken as averages over 10 separate simulations. Each timing on GIANT data set is also an average over 10 independent runs with the same matrix $L$.

does not depend on $n$ (see Figure 4, middle panel), which could be explained by the effective degrees of freedom (sparsity of the solution $x^*$) being largely unaffected by $n$.

Based on these results and the earlier finding that the effective rank remains approximately constant (Figure 3, left panel), we infer that the active set method effectively exploits the sparsity of the solution to the QP subproblem. We further note that poor conditioning of the Hessian may favor the active set method because it tends to search over sparse solutions where the reduced Hessian is better behaved. Another key benefit of the active set method is that it is able to use a good initial estimate when such an estimate is available ("warm starting"), whereas such an action is difficult to achieve with IP methods.

#### 4.2.4 Comparing mix-SQP and KWDual

Based on the numerical results above, we concluded that when both $n$ and $m$ are large, the fastest approach is `NN-SQP-A-QR`, which we have have named MIX-SQP. We compared MIX-SQP, implemented in Julia, against the `KWDual` function from the R package `REBayes` (Koenker & Gu 2017), a state-of-the-art solver that interfaces to the commercial software `MOSEK` (this is `D-IP-NA-F`). For fair comparison, all timings of `KWDual` calls from Julia were recorded in R so that communication overhead in passing variables between Julia and R was not included in the runtimes.

Although R often does not match the performance of Julia, an interactive programming language that has achieved benchmark results comparable to C++ (Bezanson et al. 2012), `KWDual` is exceptionally fast because most of the computations are performed by `MOSEK`, an industry-grade solver made available as an architecture-optimized dynamic library. Therefore, it is significant that our Julia implementation consistently outperformed `KWDual` (Figure 5). In particular, for moderately large $n$ and $m$ ($n \gtrsim 10^6, m \gtrsim 200$) MIX-SQP was almost 100 times faster than `KWDual`, never taking more than 10 seconds on average. Also observe that the `KWDual` runtimes increased more rapidly with $m$ because this optimization algorithm did not benefit from the reduced-rank matrix operations.

#### 4.2.5 Profiling adaptive shrinkage computations

An initial motivation for this work was our interest in applying a nonparametric EB method, "adaptive shrinkage," to very large data sets. These EB computations all involve three steps: (1) likelihood computation; (2) maximum-likelihood estimation of the mixture proportions; and (3) posterior computation. When we began this work, the second step, solved by `MOSEK`, was the computational bottleneck of our R implementation (Stephens et al. 2018)); see the left-hand panel in Figure 6, which reproduces the adaptive shrinkage computations in Julia (aside from `KWDual`). To verify that this bottleneck was not greatly impacted by the overhead of calling `MOSEK` from R inside function `KWDual`, we also recorded runtime estimates outputted directly by `MOSEK` (`MSK_DINF_OPTIMIZER_TIME`). We found that the overhead was at most 1.5 s, a small fraction of the total model-fitting time under any setting shown in Figure 6. (Note all timings of `KWDual` called from Julia were

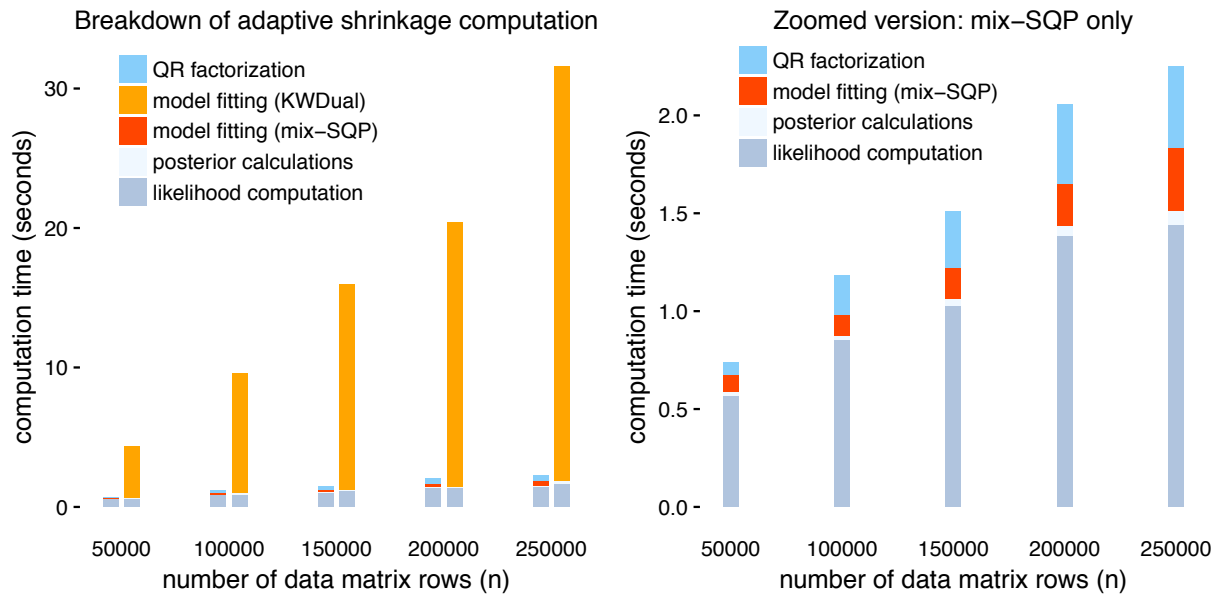


Figure 6: Breakdown of Empirical Bayes ("adaptive shrinkage") computations, in which the model-fitting (maximum-likelihood estimation) step was implemented with either MIX-SQP or `KWDual` (`MOSEK`). The adaptive shrinkage method was applied to simulated data sets with $m = 100$ and varying $n$. All runtimes were averaged over 10 independent simulations. The right-hand panel is a zoomed-in version of the MIX-SQP results shown in the left-hand plot.

recorded in R, not Julia; see Section 4.2.4.)

The model-fitting step no longer dominated the computation time after it was implemented with MIX-SQP (Figure 6). This result is remarkable considering that the likelihood calculations for the scale mixtures of Gaussians model make use of very fast probability density computations that would be difficult to improve—in both Julia and R—without introducing clever numerical approximations. (Other, more complicated models will involve more expensive likelihood computations, further increasing the fraction of time spent on the likelihood computation step.) Thus, although further improvements in optimization may be achievable (see discussion below), they would not substantially alter the total runtime of the adaptive shrinkage method. Therefore, the effort would be better spent in other areas, such as developing fast numerical approximations to the likelihood computations.

# 5 Conclusions and potential extensions

We have proposed a combination of optimization and linear algebra techniques to accelerate maximum likelihood estimation of mixture proportions. The benefits of our methods are particularly evident at settings in which the number of mixture components, $m$, is moderate (up to one thousand) and the number observations, $n$, is large. In such settings, computing the Hessian is expensive—$O(nm^2)$ effort—much more so than Cholesky factorization of the Hessian, which is $O(m^3)$. Based on this insight, we developed a sequential quadratic programming approach that makes best use of the (expensive) Hessian information, and minimizes the number of times it is calculated. We also used linear algebra techniques, specifically the RRQR factorization, to reduce the computational complexity of Hessian evaluation by exploiting the fact that matrix $L$ often has (numerically) low rank. These linear algebra improvements were possible by developing a customized SQP solver, in contrast to the use of a commercial black-box optimizer such as MOSEK. Our SQP method also benefits from the use of an active set algorithm to solve the QP subproblem, which can take advantage of sparsity in the solution vector. The overall result is that for problems with $n > 10^5$, MIX-SQP can achieve a 100-fold speedup over `KWDual`, which applies the commercial `MOSEK` interior point solver to a dual formulation of the problem.

With MIX-SQP, the optimization step is no longer the computational bottleneck in implementing nonparametric EB methods such as Koenker & Mizera (2014) and Stephens (2016). Instead, we expect that likelihood computations will dominate. If progress can be made in computing $L$, then other directions may be worth exploring. For example, noting from (15) that when $n \gg m$ evaluating the gradient is roughly $m$ times cheaper than the Hessian, *quasi-Newton methods* may be advantageous in this setting (Nocedal & Wright 2006). Quasi-Newton methods, including the most popular version, BFGS, approximate $H$ by means of a secant update that employs derivative information without ever computing the Hessian. While such methods may take many more iterations compared with exact Hessian methods, their iterations are $m$ times cheaper and, under mild conditions, also exhibit the fast superlinear convergence of Newton methods sufficiently close to the solution (Nocedal & Wright 2006).

Since $n$ is the dominant component of the computational complexity in the problem

settings we explored, another promising direction is the use of *stochastic approximation methods.* In the Newton setting, one could explore stochastic quasi-Newton (Byrd et al. 2016) or LiSSA (Agarwal et al. 2016) methods.

As we briefly mentioned in the results above, an appealing feature of SQP approaches is that they can easily be warm started. This is much more difficult for interior point methods (Potra & Wright 2000). Warm starting refers to sequential iterates of a problem becoming sufficiently similar that information about the subproblems that is normally difficult to compute from scratch ("cold") can be reused as a good initial estimate of the solution. The same idea also applies to the QP subproblems. Since, under general assumptions, the active set settles to its optimal selection before convergence, this suggests that the optimal working set $\mathcal{W}$ for subproblem $\mathcal{P}$ will often provide a good initial guess for the optimal working set $\mathcal{W}^\star$ for a similar subproblem $\mathcal{P}^\star$.

## SUPPLEMENTARY MATERIAL

**Appendix:** Description of real data, Further examples, Proof of theoretical results and algorithm routines (Appendix.pdf)

**MIX-SQP implementation:** Codes, Jupyter notebooks, and example datasets for an instruction for running the code. (mixSQP.zip)

# References


Agarwal, N., Bullins, B. & Hazan, E. (2016), 'Second-order stochastic optimization for machine learning in linear time', *arXiv* **1602.03943**.

Aharon, M., Elad, M. & Bruckstein, A. (2006), 'rmK-SVD: an algorithm for designing overcomplete dictionaries for sparse representation', *IEEE Transactions on Signal Processing* **54**(11), 4311–4322.

Andersen, E. D. & Andersen, K. D. (2000), The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm, *in* 'High performance optimization', Springer, pp. 197–232.

Atkinson, S. E. (1992), 'The performance of standard and hybrid EM algorithms for ML estimates of the normal mixture model with censoring', *Journal of Statistical Computation and Simulation* **44**(1–2), 105–115.

Auton, A. et al. (2015), 'A global reference for human genetic variation', *Nature* **526**(7571), 68–74.

Bezanson, J., Karpinski, S., Shah, V. B. & Edelman, A. (2012), 'Julia: a fast dynamic language for technical computing', *arXiv* **1209.5145**.

Brown, L. D. & Greenshtein, E. (2009), 'Nonparametric empirical Bayes and compound decision approaches to estimation of a high-dimensional vector of normal means', *Annals of Statistics* pp. 1685–1704.

Byrd, R. H., Hansen, S. L., Nocedal, J. & Singer, Y. (2016), 'A stochastic quasi-Newton method for large-scale optimization', *SIAM Journal on Optimization* **26**(2), 1008–1031.

Cordy, C. B. & Thomas, D. R. (1997), 'Deconvolution of a distribution function', *Journal of the American Statistical Association* **92**(440), 1459–1465.

Dempster, A. P., Laird, N. M. & Rubin, D. B. (1977), 'Maximum likelihood estimation from incomplete data via the EM algorithm', **39**, 1–38.

Dunning, I., Huchette, J. & Lubin, M. (2017), 'Jump: A modeling language for mathematical optimization', *SIAM Review* **59**(2), 295–320.

Golub, G. H. & Van Loan, C. F. (2012), *Matrix Computations*, Vol. 3, JHU Press.

Halko, N., Martinsson, P.-G. & Tropp, J. A. (2011), 'Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions', *SIAM Review* **53**(2), 217–288.

Ho, K. L. & Olver, S. (2018), 'LowRankApprox.jl: fast low-rank matrix approximation in Julia'.
**URL:** *https://doi.org/10.5281/zenodo.1254148*

Jiang, W. & Zhang, C.-H. (2009), 'General maximum likelihood empirical bayes estimation of normal means', *Annals of Statistics* **37**(4), 1647–1684.

Johnstone, I. M. & Silverman, B. W. (2004), 'Needles and straw in haystacks: empirical Bayes estimates of possibly sparse sequences', *Annals of Statistics* **32**(4), 1594–1649.

Kiefer, J. & Wolfowitz, J. (1956), 'Consistency of the maximum likelihood estimator in the presence of infinitely many incidental parameters', *Annals of Mathematical Statistics* pp. 887–906.

Kim, D., Sra, S. & Dhillon, I. S. (2010), 'Tackling box-constrained optimization via a new projected quasi-Newton approach', *SIAM Journal on Scientific Computing* **32**(6), 3548–3563.

Koenker, R. & Gu, J. (2017), 'REBayes: an R package for empirical Bayes mixture methods', *Journal of Statistical Software* **82**(8), 1–26.

Koenker, R. & Mizera, I. (2014), 'Convex optimization, shape constraints, compound decisions, and empirical bayes rules', *Journal of the American Statistical Association* **109**(506), 674–685.

Laird, N. (1978), 'Nonparametric maximum likelihood estimation of a mixing distribution', *Journal of the American Statistical Association* **73**(364), 805–811.

Nocedal, J. & Wright, S. J. (2006), *Nonlinear Optimization*, Springer.

Pearson, K. (1894), 'Contributions to the mathematical theory of evolution', *Philosophical Transactions of the Royal Society of London. A* **185**, 71–110.

Potra, F. A. & Wright, S. J. (2000), 'Interior-point methods', *Journal of Computational and Applied Mathematics* **124**(1–2), 281–302.

Redner, R. A. & Walker, H. F. (1984), 'Mixture densities, maximum likelihood and the EM algorithm', *SIAM Review* **26**(2), 195–239.

Salakhutdinov, R., Roweis, S. & Ghahramani, Z. (2003), Optimization with EM and expectation-conjugate-gradient, *in* 'Proceedings of the 20th International Conference on Machine Learning', pp. 672–679.

Stephens, M. (2016), 'False discovery rates: a new deal', *Biostatistics* **18**(2), 275–294.

Stephens, M., Carbonetto, P., Dai, C., Gerard, D., Lu, M., Sun, L., Willwerscheid, J., Xiao, N. & Zeng, M. (2018), *ashr: Methods for Adaptive Shrinkage, using Empirical Bayes*. **URL:** *http://CRAN.R-project.org/package=ashr*

Varadhan, R. & Roland, C. (2008), 'Simple and globally convergent methods for accelerating the convergence of any EM algorithm', *Scandinavian Journal of Statistics* **35**(2), 335–353.

Wood, A. R., Esko, T., Yang, J., Vedantam, S., Pers, T. H. et al. (2014), 'Defining the role of common variation in the genomic and biological architecture of adult human height', *Nature Genetics* **46**(11), 1173–1186.

Wright, S. J. (1998), 'Superlinear convergence of a stabilized sqp method to a degenerate solution', *Computational Optimization and Applications* **11**(3), 253–275.

# A Active set method details

Algorithm 2 provides a brief pseudocode of the active set method MIX-ACTIVE-SET we implement and use in our main algorithmic loop, Algorithm 1. See also (Nocedal & Wright 2006, §16.5). In short, we iteratively solve an equality-constrained subproblem:

$$q^{(l)} = \arg\min_{q} \tfrac{1}{2} q^T H_t q + q^T b_l \quad \text{subject to} \quad q_i = 0, \quad \forall i \in \mathcal{W}^{(l)}, \tag{21}$$

which will be called the active set subproblem.

Let $H$ and $g$ be the Hessian and gradient at the current outer iterate $x$ computed before the active set method starts. At the $l$th iteration, we compute $b^{(l)} = Hy^{(l)} + 2g - \mathbb{1}_m$, solve (21), and take $y^{(l+1)} = y^{(l)} + \eta_l q_l$ where $\eta_l \in [0, 1]$ is the largest value that maintains feasibility with respect to the inequality constraints. If $\eta_l = 1$, then there are no blocking constraints. We then move on to the next iterate $l = l + 1$. If not, $\eta_l$ is blocked by some constraint, which should be added to the active working set.

If we reach a quadratic minimizer in the current working set, then we check whether it is a solution to the QP subproblem via the usual Karush-Kuhn-Tucker (KKT) conditions of (17). If the KKT conditions are not satisfied, then there must exist a Lagrange multiplier that is negative, so the co-ordinate corresponding to the minimum multiplier is removed from $\mathcal{W}^{(t)}$. If the KKT conditions are satisfied, we finish solving the active set subproblem with returning $y^{(l)}$ as a solution.

More details can be found in the code provided in the supplementary material.

# B GIANT data processing details

We retrieved file `GIANT_HEIGHT_Wood_et_al_2014_publicrelease_HapMapCeuFreq.txt.gz` from the GIANT project Wiki (http://portals.broadinstitute.org/collaboration/giant). The original tab-delimited text file contains summary statistics—regression coefficient estimates $z_j$ and their standard errors $s_j$ (columns "b" and "SE" in the text file)—for 2,550,858 SNPs $j$ on chromosomes 1–22 and X. These summary statistics were computed from a meta-analysis of 79 genome-wide association studies of human height; see Wood

**Algorithm 2:** MIX-ACTIVE-SET: active set method for the QP subproblem

**Input:** $g \in \mathbb{R}^k$, $H \in \mathbb{R}^{k\times k}$, Optimization tolerance $\epsilon$
Sparse initial point $y^{(0)} = x$, $\mathcal{W}^{(0)} = [m]\backslash \text{supp}(y^{(0)})$, $l = 0$

**while** *not converged* **do**

- `/* solve (18) on the current working set */`
- $b^{(l)} \leftarrow Hy^{(l)} + 2g - \mathbb{1}_m$ `/* Lemma 3.4 */`
- $q^{(l)} \leftarrow \underset{\text{supp}(q)\subset[m]\backslash\mathcal{W}^{(l)}}{\arg\min} \frac{1}{2}q^T Hq + q^T b^{(l)}$
- `/* if reaches at minimizer on the current working set */`
- **if** $\|q^{(l)}\|_2 < \epsilon$ **then**
  - `/* check convergence */`
  - **if** $\min_k b_k^{(l)} > -\epsilon$ **then**
    - break `/* this is a global solution */`
  - **else**
    - $i \leftarrow \arg\min_k b_k^{(l)}$
    - $\mathcal{W}^{(l+1)} \leftarrow \mathcal{W}^{(l)}\backslash\{i\}$ `/* remove index with smallest multiplier */`
- **else**
  - **if** $y^{(l)} + q^{(l)} \succeq 0$ **then**
    - $y^{(l+1)} \leftarrow y^{(l)} + q^{(l)}$
  - **else**
    - `/* if there is a blocking constraint */`
    - $\eta_l \leftarrow \max\{\eta : y^{(l)} + \eta q^{(l)} \succeq 0\}$
    - $i \leftarrow \text{find}(y_i^{(l)} + \eta_l q_i^{(l)} = 0)$
    - $\mathcal{W}^{(l+1)} \leftarrow \mathcal{W}^{(l)} \cup \{i\}$ `/* add blocking index */`
    - $y^{(l+1)} \leftarrow y^{(l)} + \eta_l q^{(l)}$
- $l \leftarrow l + 1$

**return** MIX-ACTIVE-SET$(x, g, H) = y^{(l)}$

et al. (2014) for details about the studies and meta-analysis methods used. We removed 39,812 SNPs that were not identified in 1000 Genomes Project, Phase 3 (Auton et al. 2015), an additional 384,254 SNPs where the coding strand was ambiguous, and 114 more SNPs with alleles that did not match the 1000 Genomes data, for a final data set containing $n = 2{,}126{,}678$ SNPs. (Note that the signs of the $z_j$ estimates were flipped when necessary to align with the 1000 Genomes SNP genotype encodings, although in practice this should have no effect on our results here since the prior is a mixture of zero-centered normals.) The processed GIANT data are included with the git repository accompanying this paper.

# C Further examples

Several frequently encountered problems fit our paradigm. The main modeling choice of these examples is that of dictionary $\mathcal{G}$, as the function $\varphi(\cdot;\cdot,\cdot)$ entering the definition of the problem data in (2) (and originating from (5)) corresponds to the normal distribution.

**Example C.1** (Adaptive Shrinkage)**.** *Given a fine grid of component variances $\sigma_1^2, \cdots, \sigma_m^2$, we consider a class $\mathcal{G}$ of all scale mixture of $m$ Gaussian distributions around 0.*

$$g = \sum_{k=1}^{m} x_k N(0, \sigma_k^2), \quad x \in \mathcal{S}^m$$

*That is, the dictionary $\mathcal{G}$ is generated by the densities $g_k = N(0, \sigma_k^2)$ with known variance $\sigma_k^2$, the latter chosen on a fine mesh. Stephens (2016), in the context of the R* `ashr` *package, considers other classes of dictionary functions such as scale mixture of normal distributions and uniform distributions. Those can also easily be included in this framework.*

**Example C.2** (Generalized Maximum Likelihood)**.** *Given a fine grid of mean values $\mu_1, \cdots, \mu_m$, we consider a class $\mathcal{G}$ of location-indexed mixtures*

$$g = \sum_{k=1}^{m} x_k N(\mu_k, 1), \quad x \in \mathcal{S}^m.$$

*Jiang & Zhang (2009) and Koenker & Mizera (2014) study nonparametric versions of this problem as also known as the compound decision or needles and straws in haystack problem. For more details, see Johnstone & Silverman (2004) and Brown & Greenshtein (2009).*

**Example C.3** (Shape-constrained Density Estimation)**.** *Koenker & Mizera (2014) considered a shape-constrained density estimation. One chooses a fine grid of points $y_1 < y_2 < \ldots < y_n$ over the data range, and the (marginal) mixture density at $y_j$ is $e^{z_j}$, where $z_j$ is a real number. Under the strong log-concavity constraint on the marginal log-likelihood, one gets the following dual problem:*

$$\begin{aligned} \text{minimize} \quad & h(Ax+b) + \sum_{k=1}^{m} x_k \\ \text{subject to} \quad & x \in \mathbb{R}_+^m, \end{aligned} \tag{22}$$

*where $h$ is a weighted entropy function; that is, $h(y) = \frac{1}{n}\sum_{i=1}^{n} c_i y_i \log y_i$ so that $f(x) = h(Ax+b)$ matches with our framework.*

# D Proof of theoretical eesults

This section is dedicated to proving the theoretical results in the paper.

## D.1 Proof of Theorem 3.2

First note that because of the monotonicity and unboundedness of the objective over the positive orthant, the solution to (9) is preserved if we relax the simplex constraint $x \in \mathcal{S}^m = \{x : \mathbb{1}^T x = 1, x \succeq 0\}$ to a system of linear inequality constraints as $x \in \{x : \mathbb{1}^T x \leq 1, x \succeq 0\}$. Slater's condition is trivially satisfied for both formulations of the simplex constraints, and the feasible set is compact. The solution then satisfies the KKT optimality conditions; *i.e.*, at the solution $x^*$ there exists a $\lambda^* \geq 0$ and a $\mu^* \succeq 0$ such that

$$\nabla\phi(x^*) + \lambda^* \mathbb{1}_m - \mu^* = 0, \quad (\mu^*)^T x^* = 0, \quad \mathbb{1}_m^T x^* = 1.$$

We therefore conclude that (9) is equivalent to

$$\arg\min_{x \in \mathbb{R}_+^m} \left[\phi(x) + \lambda^* \left(\textstyle\sum_{k=1}^m x_k - 1\right)\right], \tag{23}$$

where $\mathcal{R}_+^m = \{x \in \mathbb{R}^m : x \succeq 0\}$ is the $m$-dimensional positive orthant. We claim that for any $\lambda > 0$, the solution of the Lagrange relaxation of the problem

$$x^*(\lambda) = \arg\min_{x \in \mathbb{R}_+^m} \left[\phi(x) + \lambda \sum_{k=1}^m x_k\right]$$

is the solution of the original problem multiplied by a constant, $\lambda/\lambda^*$. This occurs because

$$\begin{aligned} x^*(\lambda) &= \operatorname*{arg\,min}_{x\in\mathbb{R}^m_+} \left[\phi(x) + \lambda \sum_{k=1}^m x_k\right] \\ &= \operatorname*{arg\,min}_{x\in\mathbb{R}^m_+} \left[\phi((\lambda/\lambda^*)x) + \lambda^* \sum_{k=1}^m (\lambda/\lambda^*) x_k\right] \\ &= \frac{\lambda^*}{\lambda}\left(\operatorname*{arg\,min}_{x'\in\mathbb{R}^m_+} \left[\phi(x') + \lambda^* \sum_{k=1}^m x'_k\right]\right) = \frac{\lambda^*}{\lambda} x^*(\lambda^*) \end{aligned}$$

provided that $\lambda^* > 0$. Here the second equality follows from the scale-free assumption on $\phi(x)$, which in turn implies that $\arg\min[\phi(x) + \psi(x)] = \arg\min[\phi(cx) + \psi(x)]$ for any $c > 0$. Note that at a solution of (9), we cannot have $\lambda^* = 0$. If we do, then the point $x^*$ satisfies the KKT conditions of the problem where the constraints $\sum_{i=1}^m x_k = 1$ are removed, and it must then be a solution of that problem. Since the objective function decreases as we scale up $x$, such problems clearly are unbounded below and thus cannot have an optimal solution (as scaling $x$ up keeps decreasing the function value while preserving nonnegativity of entries in $x$). Since the solution of (9) must satisfy $\sum_{k=1}^m x_k = 1$, the conclusion follows.

## D.2 Proof of Corollary 3.3

Consider the KKT conditions for the optimality again: at the solution $x^*$ there exists $\lambda^* \geq 0$ and $\mu^* \succeq 0$ such that

$$\nabla f(x^*) + \lambda^* \mathbb{1}_m - \mu^* = 0, \quad (\mu^*)^T x^* = 0, \quad \mathbb{1}_m^T x^* = 1. \tag{24}$$

By multiplying $(x^*)^T$ to the first equality, we have

$$(x^*)^T \nabla f(x^*) + \lambda^* = 0 \iff \lambda^* = 1, \tag{25}$$

since $\nabla f(x) = -\frac{1}{n} L^T d$ and $d_i = (Lx)_i^{-1}$ by Lemma 3.4.