Compressed and Penalized Linear Regression

Darren Homrighausen*
Department of Statistics, Colorado State University
and
Daniel J. McDonald
Department of Statistics, Indiana University

July 29, 2019

Abstract

Modern applications require methods that are computationally feasible on large datasets while retaining good statistical properties. Recent work has focused on developing fast and randomized approximations for solving least squares problems when the data are too large to fit into memory easily or when computations are at a premium. Many of these techniques rely on data-driven subsampling or random compression. In this paper, we provide new approximate algorithms for solving penalized least squares problems which have improved statistical performance relative to existing methods. We provide the first efficient methods for tuning parameter selection, compare our methods with current approaches via simulation and application, and provide theoretical intuition which makes explicit the impact of approximation on statistical efficiency and demonstrates the necessity of careful parameter tuning.

Keywords: preconditioning; sketching; regularization; ordinary least squares; ridge regression

*Darren Homrighausen is currently Visiting Assistant Professor at Southern Methodist University. The authors gratefully acknowledge support from the National Science Foundation (grant DMS-1407543 to DH; grants DMS-1407439 and DMS–1753171 to DJM) and the Institute for New Economic Thinking (grant INO14–00020 to DH and DJM).

1 Introduction

Recent work in large-scale data analysis has focused on developing fast and randomized approximations to important numerical linear algebra tasks such as solving least squares problems (Becker et al., 2017; Pilanci and Wainwright, 2015; Wang et al., 2017; Zhang et al., 2013) and finding spectral decompositions (Gittens and Mahoney, 2013; Halko et al., 2011; Homrighausen and McDonald, 2016). These approaches, known as *compression*, *sketching*, or *preconditioning*, take a given data set and construct smaller, "compressed," data. This compressed data is then used as a surrogate for the original data, so that statistical analyses can be performed more quickly or use less storage space. For example, NOAA satellites with Advanced Very High Resolution Radiometers are unable to store all the data they collect each day. Thus global coverage data—as opposed to high resolution local coverage data, which is immediately offloaded—is downsampled from the original 1 kilometer resolution to 4 km before being sent to a ground station (Frey et al., 1996; Staten et al., 2016). Hence, all statistical models based on these measurements are necessarily using compressed data. Of course, analyses using these subsampled data may be less accurate than if they had used all the data, but the analyses can also be performed much more quickly.

Existing theoretical justifications for these sorts of approximations upper bound the difference in some objective function (say the sum of squared residuals) evaluated at the "compressed" solution relative to the full-data solution. In this paper, we take a more statistical perspective: we investigate the performance in terms of parameter estimation and prediction risk, integrating over the randomness in the data as well as any in the compression algorithm. Leveraging insights into the statistical behavior of the most commonly used compressions, we develop and explore novel algorithms for compressed least squares and provide principled methods of tuning parameter selection, an important aspect of statistical performance absent from existing literature.

As a preview of the benefits or our methods, Figure 1 compares the computational timing and out-of-sample prediction error of four compression methods relative to ridge regression on the whole data. The figure shows two existing methods of compression—full compression (FC) and partial compression (PC)—and the two methods we develop here—linear combination compression (linear) and convex combination compression (convex)—evaluated on

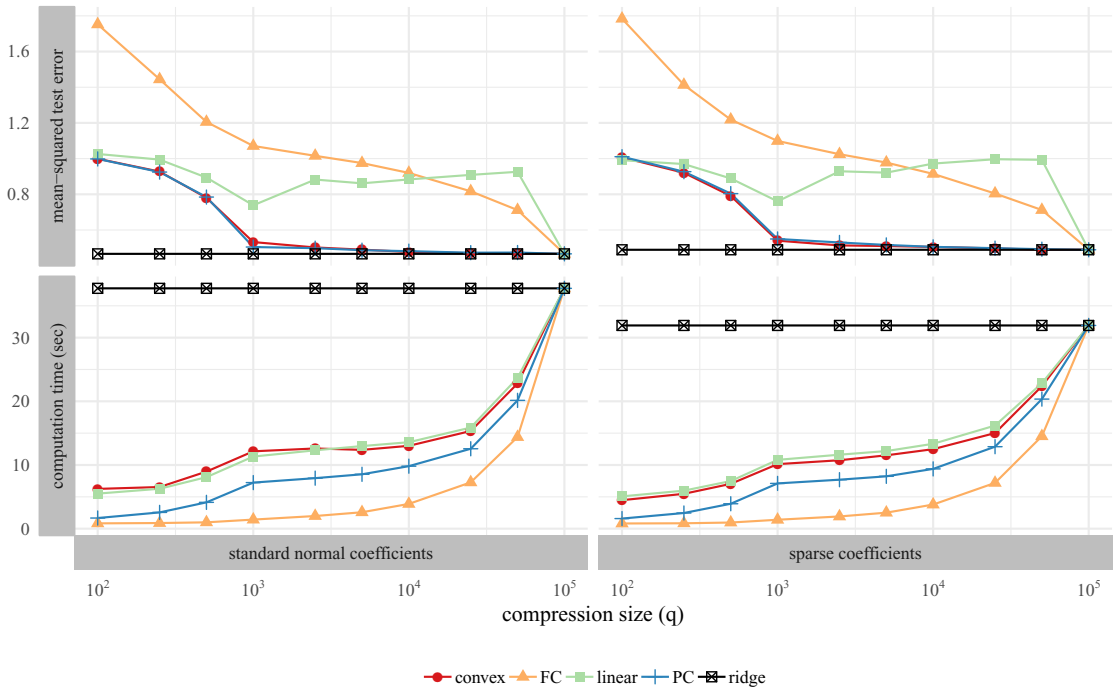

Figure 1: These plots show the computational time and predictive accuracy of two existing methods (FC and PC) and our two proposed methods (convex and linear) as compared to performing ridge regression on the whole dataset.

data generated from both a dense and a sparse linear model. The parameter q determines the amount of compression with lower values meaning more compression. Heuristically, one can think of sampling q observations from the original data to use for analysis. In this example, the simulated data has $n = 100{,}000$ observations and $p = 1000$ predictors. Even when $q < p \ll n$, the accuracy lost by these methods is small relative to the increase in computation time, and essentially negligible for $q = p$.

1.1 Overview of the problem

Suppose we make n paired, observations $X_i \in \mathbb{R}^p$ and $Y_i \in \mathbb{R}$, $i = 1, \ldots, n$, where X_i is a vector of measurements, Y_i is the associated response, and $n \gg p$ with both n and p very large. Concatenating the vectors X_i row-wise into a matrix $X := [X_1^\top, \ldots, X_n^\top]^\top \in \mathbb{R}^{n \times p}$ and the responses Y_i into a vector Y, we assume that there exists a $\beta_* \in \mathbb{R}^p$ such that

$$Y = X\beta_* + \sigma\varepsilon,$$

with $\mathbb{E}[\varepsilon] = 0$. For our theoretical results in Section 6, we will assume that $\mathbb{V}[\varepsilon] = I_n$ for convenience, but this assumption can be removed with some care and is immaterial for the methodological development.

One can seek to estimate β via linear regression. That is, for any vector $\beta \in \mathbb{R}^p$, we define $\ell(\beta) := \sum_{i=1}^n (Y_i - X_i^\top \beta)^2$ and, writing the (squared) Euclidean norm as $||X\beta - Y||_2^2 := \sum_{i=1}^n (Y_i - X_i^\top \beta)^2$, a least squares solution is a vector $\widehat{\beta} \in \mathbb{R}^p$ such that

$$\ell(\widehat{\beta}) = \min_{\beta} ||X\beta - Y||_2^2 . \tag{1}$$

This is given by $\widehat{\beta} = X^\dagger Y$, where $X^\dagger$ is the Moore-Penrose pseudo inverse of X. If X has full column rank, the solution simplifies to $\widehat{\beta} = (X^\top X)^{-1} X^\top Y$.

The ordinary least squares (OLS) solution can be found stably in $O(np^2)$ computations when $p > n$ using, for example, routines in LAPACK such as the QR decomposition, Cholesky decomposition of the normal equations, or the singular value decomposition (Golub and Van Loan, 2012). It also has a few well-known statistical properties such as being a minimum variance unbiased estimator and the best linear unbiased estimator. However, the classic numerical linear algebraic techniques that compute $\widehat{\beta}$ require loading X and Y into random access memory, so this computation can be infeasible or undesirable in practice.

The big-data regime we consider, i.e. $n \gg p$ and both very large, can happen in many different scientific areas such as psychology, where cellular phones are used to collect high-frequency data on individual actions; atmospheric science, where multiresolution satellite images are used to understand climate change and predict future weather patterns; technology companies, which use massive customer databases to predict tastes and preferences; or medical imaging, where multiple high-resolution brain scans are acquired over some period of time.

1.2 Prior work

A very popular approach in the approximation literature (Drineas et al., 2011; Rokhlin and Tygert, 2008; Woodruff, 2014) is to generalize Equation (1) to include a compression matrix $Q \in \mathbb{R}^{q \times n}$, with $n > q > p$, $\ell_Q(\beta) = ||Q(X\beta - Y)||_2^2$. The associated approximation to $\widehat{\beta}$ is

the *fully compressed estimator* (FC)

$$\widehat{\beta}_{FC} = \underset{\beta}{\operatorname{argmin}} \, ||Q(X\beta - Y)||_2^2 \,, \tag{2}$$

which can be computed via standard techniques by substituting the compressed data QX and QY for the original. We defer discussion of strategies and trade-offs for specific choices of Q to Section 2.4, but fast matrix multiplication will require some structure on Q or multithreaded implementations.

Standard theoretical justifications for full compression define a tolerance ϵ and a compression parameter $q = q(\epsilon)$ such that, with high probability (Drineas et al., 2012, 2011),

$$\ell(\widehat{\beta}_{FC}) \leq (1+\epsilon)\ell(\widehat{\beta}). \tag{3}$$

In this case, the probability is stated with respect to the process that generates Q only while the data are considered fixed. Thus (3) is a worst-case analysis since it must hold uniformly over all data sets, regardless of the "true" data generating process (see results in Blendenpik (Avron et al., 2010) or LSRN (Meng et al., 2014) for practical considerations).

Relative to the computational properties of these compression methods, there has been comparatively little work on their statistical properties. Raskutti and Mahoney (2015) analyze various relative efficiency measures of $\widehat{\beta}_{FC}$ versus $\widehat{\beta}$ as a function of the compression matrix Q. They find that the statistical quality of $\widehat{\beta}_{FC}$ depends on the oblique projection matrix $U(QU)^{\dagger}U$, where U is the left singular matrix of X. Additionally, Zhou et al. (2009) consider this fully compressed model with the addition of the LASSO penalty under a variety of metrics like variable selection consistency and predictive risk consistency. Ma et al. (2015) examine mean-squared error performance for leverage-score compression of ordinary least squares. Their theoretical results are most similar to ours, though for different methodologies. Pilanci and Wainwright (2015) examine the algorithmic and statistical properties of a related method, defined below as *partial compression* (they call it "Hessian sketching").

As illustrated in Figure 1, compression techniques operate if $q < p$ (though without well-developed theoretical guarantees) and also in high dimensions ($p > q > n$), but this situation is not well-studied. Compressing instead as $XQ\beta - Y$ is sometimes termed *random projection*. Zhang et al. (2013) proposed "dual random projection" for this context and Wang et al. (2017) show how to sketch and apply random projection simultaneously (with

$Q_1(XQ_2\beta - Y)$). However, the intuition behind random projection requires a statistical model under which X is low rank, and none of this work examines the statistical properties (only the algorithmic ones). Statistically, it is more reasonable to use the low-rank structure rather than operate randomly. A good example of such an approach is the work on supervised principal components analysis (Bair et al., 2006; Ding and McDonald, 2017; Paul et al., 2008).

1.3 Our contributions

This paper, in contrast with most previous research, adopts the perspective that approximations mimicking the least squares estimator $\widehat{\beta}$ may produce faster methods, but may not result in good estimators. Instead, we seek approximations that minimize estimation and/or prediction error (by analyzing quantities integrated over the data) while simultaneously decreasing computational burdens. Our goal is not to beat full-data methods but to create approximation methods with better statistical performance than existing approaches.

We argue that $\widehat{\beta}_{FC}$, like $\widehat{\beta}$, is unbiased for β and hence must have a larger estimation and prediction error than a regularized version by the Gauss-Markov theorem. The alternative, namely *partial compression* ($\widehat{\beta}_{PC}$ defined in Equation (5) below), is biased without any additional regularization, and often (though not always) performs well in practice. Therefore, we propose to combine the two and add regularization for improved mean squared error performance. This improvement is achieved by introducing a tuning parameter to directly calibrate bias and variance. In this paper, we use an ℓ_2 penalty, because we can then derive statistical results that are comparable to others in the literature. Other methods, however, such as ℓ_1, Dantzig Selector, bridge, or the nonnegative garrote could all be used. Our estimators can be computed for a fraction of the cost of OLS or ridge regression while improving performance relative to existing compressed alternatives. Our main contributions are as follows:

- We present a new family of regularized estimators which linearly combine $\widehat{\beta}_{FC}$ and the *partially compressed* estimator and add regularization (Section 2).
- We derive an unbiased risk estimator for selecting tuning parameters in compressed, regularized problems and demonstrate how to calculate the entire solution path without

increasing the order of computations (Section 3). There is no previous work which examines the effect of tuning parameter selection when combined with compression.

- We examine our regularized estimators in simulated and real data examples and find that they lead to improved performance relative to existing techniques (Section 4 and Section 5).

- We give theoretical results establishing the mean squared error performance of ℓ_2-regularized compressed estimators for random compression matrices (Section 6). These results show explicitly the relationship between the amount and type of compression and the statistical properties of the procedure.

2 Compressed regression

This section presents the standard compressed least squares regression estimators, introduces our modifications, and discusses the specific form of the compression matrix Q we consider.

2.1 Compressed least squares regression

The fully compressed least squares estimator (notated later as FC), defined in Equation (2), is the solution to

$$\min_{\beta} ||Q(X\beta - Y)||_2^2 = \min_{\beta} \left(\beta^\top X^\top Q^\top Q X\beta - 2\beta^\top X^\top Q^\top QY\right). \tag{4}$$

An alternative called *partial compression* (PC) (Becker et al., 2017; Pilanci and Wainwright, 2015) is the solution to

$$\min_{\beta} \left(\beta^\top X^\top Q^\top Q X\beta - 2\beta^\top X^\top Y\right), \tag{5}$$

which removes the compression matrix from the "$X^\top Y$" term. Depending on the form of the compression matrix Q, there may not be unique solutions to Equations (4) or (5).

2.2 Compressed ridge regression

A well used technique to stabilize the least-squares problem is ridge regression (Hoerl and Kennard, 1970). The ridge regression problem can be written in the Lagrangian form as

$$\widehat{\beta}(\lambda) := \underset{\beta}{\operatorname{argmin}} \, ||X\beta - Y||_2^2 + \lambda \, ||\beta||_2^2 \,. \tag{6}$$

While $\widehat{\beta}(\lambda)$ has better numerical stability than $\widehat{\beta}(0) \equiv \widehat{\beta}$, it improves neither the computational complexity, which is still $O(np^2)$, nor the storage, which is $O(np)$. Besides being more numerically stable, there exists $\lambda > 0$ such that $\widehat{\beta}(\lambda)$ has lower risk than $\widehat{\beta}$. These results beg the question: if $\widehat{\beta}(\lambda)$ is a better overall procedure, why not compress it instead? Leveraging this insight, we define the *fully compressed ridge estimator*, in analogue to Equation (4), as

$$\widehat{\beta}_{FC}(\lambda) = \underset{\beta}{\operatorname{argmin}} \, ||Q(X\beta - Y)||_2^2 + \lambda \, ||\beta||_2^2 = (X^\top Q^\top Q X + \lambda I)^{-1} X^\top Q^\top Q Y. \tag{7}$$

Likewise, analogous to Equation (5), the *partially compressed ridge estimator* is

$$\widehat{\beta}_{PC}(\lambda) = \underset{\beta}{\operatorname{argmin}} \left(\beta^\top X^\top Q^\top Q X \beta - 2\beta^\top X^\top Y + \lambda \, ||\beta||_2^2\right) = (X^\top Q^\top Q X + \lambda I)^{-1} X^\top Y.$$

Ignoring numerical issues for very small λ, both of these estimators always have a unique solution regardless of Q and X.

2.3 Linear combination compressed ridge regression

Standard linear model theory gives that $Y = X\widehat{\beta} + \widehat{e}$ where $\mathbb{P}(\widehat{e} \in \operatorname{col}(X)) = 0$, and $\mathbb{E}\left[\widehat{e}\right] = 0$. Therefore,

$$\begin{aligned} \widehat{\beta}_{FC}(0) &= (X^\top Q^\top Q X)^\dagger X^\top Q^\top Q Y = (X^\top Q^\top Q X)^\dagger X^\top Q^\top Q (X\widehat{\beta} + \widehat{e}) \\ &= \widehat{\beta} + (X^\top Q^\top Q X)^\dagger X^\top Q^\top Q \widehat{e}, \end{aligned}$$

and hence $\mathbb{E}\left[\widehat{\beta}_{FC}(0)\right] = \mathbb{E}\left[\widehat{\beta}\right] = \beta_*$. This indicates that, like $\widehat{\beta}$, $\widehat{\beta}_{FC}(0)$ is unbiased and hence must have larger variance than $\widehat{\beta}$. On the other hand,

$$\widehat{\beta}_{PC}(0) = (X^\top Q^\top Q X)^\dagger X^\top Y = (X^\top Q^\top Q X)^\dagger X^\top (X\widehat{\beta} + \widehat{e}) = (X^\top Q^\top Q X)^\dagger X^\top X \widehat{\beta}.$$

Hence, $\widehat{\beta}_{PC}(0)$ is a biased estimator. In an effort to reduce out-of-sample MSE by balancing bias and variance, it is reasonable to combine $\widehat{\beta}_{FC}(\lambda)$ and $\widehat{\beta}_{PC}(\lambda)$ instead of choosing between them. For this reason, we consider two estimators generated by a linear combination

$$\widehat{\beta}_{\alpha}(\lambda) = \alpha_{FC}\widehat{\beta}_{FC}(\lambda) + \alpha_{PC}\widehat{\beta}_{PC}(\lambda),$$

where $\alpha := [\alpha_{FC},\ \alpha_{PC}]$. A data-driven value $\widehat{\alpha}$ can be computed by forming the matrix $B(\lambda) = \left[\widehat{\beta}_{FC}(\lambda),\ \widehat{\beta}_{PC}(\lambda)\right] \in \mathbb{R}^{p\times 2}$, a column-wise concatenation of $\widehat{\beta}_{FC}(\lambda)$ and $\widehat{\beta}_{PC}(\lambda)$, and then solving

$$\widehat{\alpha}(\lambda) = \underset{\alpha\in\mathbb{R}^2}{\operatorname{argmin}} \left|\left| XB(\lambda) \begin{bmatrix} \alpha_{FC} \\ \alpha_{PC} \end{bmatrix} - Y \right|\right|_2^2 .$$

We refer to this estimator as *linear combination compression*. We also consider the *convex combination*: $\widehat{\alpha}(\lambda) = \operatorname{argmin}_{\alpha\in C} ||XB(\lambda)\alpha - Y||_2^2$ where $C = \left\{\alpha \in [0,1]^2 : 1^\top\alpha = 1\right\}$. We emphasize with the notation $\widehat{\alpha}(\lambda)$ that $\widehat{\alpha}$ is a deterministic function of λ (as well as B, X, Y) and not a separate tuning parameter. We will suppress its dependence on λ and simply use $\widehat{\alpha}$ in what follows. We discuss how to select λ in Section 3. For either combined estimator, an estimator of β, or a prediction $\widehat{Y}$, can be produced with $\widehat{\beta}_{\widehat{\alpha}}(\lambda) := B(\lambda)\widehat{\alpha}$ and $\widehat{Y}_{\widehat{\alpha}}(\lambda) := X\widehat{\beta}_{\widehat{\alpha}}(\lambda)$, respectively.

2.4 Compression matrices

The effectiveness of compression depends on q, the nature of Q, and the structure of X and β_*. For arbitrary Q, the multiplication QX would take $O(qnp)$ operations and, hence, could be as expensive as solving the original least squares problem. However, this multiplication is "embarrassingly parallel" (say, by the map-reduce framework) rendering the multiplication cost somewhat meaningless in contrast to the least squares solution, which is not embarrassingly parallel. Specifically, by generating one row Q_i in a sparse manner, performing the multiplication Q_iX, and then throwing away the row, we avoid ever creating or storing the $q \times n$ matrix Q and can perform each row multiplication on a different processor before recombining. Therefore, the limiting computation for solving (7) is only $O(qp^2)$.[1]

[1]Throughout this paper, our timing measurements exclude the time required to perform the compression. As discussed here, the compression operation itself is easily parallelized, but also highly system dependent.

The structure of Q is chosen, typically, either for its theoretical or computational properties. Examples are standard Gaussian entries, producing dense but theoretically convenient Q, fast Johnson-Lindenstrauss methods, or the counting sketch. A thorough discussion of these methods is outside the scope of this paper. Instead, we use a "sparse Bernoulli" matrix (Achlioptas, 2003; Dasgupta et al., 2010; Kane and Nelson, 2014; Woodruff, 2014). Here, the entries of Q are generated independently where $\mathbb{P}(Q_{ij} = 0) = 1 - 1/s$ and $\mathbb{P}(Q_{ij} = -1) = \mathbb{P}(Q_{ij} = 1) = 1/(2s)$ for some $s \geq 1$. Then, Q has approximately qn/s non-zero entries and can be multiplied quickly with high probability, while Equation (4) can be solved without parallelization in $O(qnp/s + qp^2)$ time on average. Throughout this paper, we assume Q is renormalized so that $\mathbb{E}[Q^\top Q] = I_n$.[2]

There is an important tradeoff between q and s. Larger s means that the matrix multiplication can be performed more quickly at the expense of increasing the variance of the resulting estimator (see the theoretical results in Section 6.1). Essentially, computations increase linearly in q/s while the variance decreases linearly in q/s. Thus, the tradeoff between computational burden and statistical efficiency can be made explicit for sparse Bernoulli Q. For the Johnson-Lindenstrauss transform, the matrix multiplication will be essentially computed via the Fast Fourier Transform and requires $O(np \log(q))$ operations. Counting sketches sample q rows from X according to some distribution. For more details on computational costs of these techniques, see for example Ailon and Chazelle (2006); Ma et al. (2015); Wang et al. (2017).

3 Tuning Parameter Selection

Employing a ridge penalty or other form of regularization requires appropriate selection of λ to achieve good statistical performance. Presumably, if computations or storage are at a premium, computer-intensive resampling methods such as cross-validation are unavailable.

For instance, one could take advantage of multithreading (on a quad-core CPU) or use the GPU if it is available. Many R packages exist that automatically take advantage of these possibilities easily. Therefore, the best implementation varies widely across operating systems and even across, say, Apple MacBook Pros built in the same year running the same OS. For this reason, rather than report timings for code specific to the system we used, we report timings achievable for most people using a single processor.

[2]We use a subscript on the identity matrix to denote its dimension when not immediately clear.

Therefore, we develop methods that rely on a corrected training-error estimate of the risk. These corrections depend crucially on the degrees of freedom. Specifically, the *degrees of freedom* (Efron, 1986) of a procedure $g : \mathbb{R}^n \to \mathbb{R}^n$ that produces predictions $g(Y) = \widehat{Y}$ is

$$\mathrm{df}(g) := \frac{1}{\sigma^2} \sum_{i=1}^{n} \mathrm{Cov}(g_i(Y), Y_i),$$

where $\sigma^2 = \mathbb{V}(Y_i)$.

If the response vector is distributed according to the homoskedastic model $Y \sim (\mu, \sigma^2 I)$, then we can decompose the prediction risk of the procedure g as

$$\mathrm{Risk}(g) = \mathbb{E}\,||g(Y) - \mu||_2^2 = \mathbb{E}\,||g(Y) - Y||_2^2 - n\sigma^2 + 2\sigma^2 \mathrm{df}(g). \tag{8}$$

A plug-in estimate of $\mathrm{Risk}(g)$ (analogous to C_p, Mallows, 1973) is then

$$\widehat{\mathrm{Risk}}(g) = ||g(Y) - Y||_2^2 - n\widehat{\sigma}^2 + 2\widehat{\sigma}^2 \widehat{\mathrm{df}}(g),$$

where $\widehat{\mathrm{df}}(g)$ and $\widehat{\sigma}^2$ are estimates of $\mathrm{df}(g)$ and σ^2, respectively.[3] We discuss strategies for forming $\widehat{\mathrm{df}}(g)$ in Section 3.1. As for the variance, ordinarily one would use the unbiased estimator $\widehat{\sigma}^2 = (n - \mathrm{df}(g))^{-1}\,||(I_n - \Pi_X)Y||_2^2$, where $\Pi_X = X(X^\top X)^\dagger X^\top = UU^\top$ is the orthogonal projection onto the column space of X. However, computing $I_n - \Pi_X$ is just as expensive as computing the least squares solution $\Pi_X Y$ itself. Therefore, to avoid estimating σ^2, we use generalized cross validation (Golub et al., 1979)

$$\mathrm{GCV}(g) = \frac{\frac{1}{n}\,||g(Y) - Y||_2^2}{(1 - \mathrm{df}(g)/n)^2}.$$

3.1 Estimating the degrees of freedom

For any procedure g which is linear in Y (that is, there exists some matrix Φ which does not depend on Y such that $g(Y) = \Phi Y$), $\mathrm{df}(g) = \mathrm{tr}(\Phi)$, the trace of Φ. Therefore, computing the exact degrees of freedom for $\widehat{\beta}_\alpha(\lambda)$ (that is, the linear or convex combination estimator with a fixed α) is straightforward. In this case,

$$X\widehat{\beta}_\alpha(\lambda) = XB(\lambda)\alpha = \alpha_{FC} X\widehat{\beta}_{FC}(\lambda) + \alpha_{PC} X\widehat{\beta}_{PC}(\lambda) =: \alpha_{FC} H_{FC} Y + \alpha_{PC} H_{PC} Y = \Phi Y$$

[3] See the Supplementary Material for a short derivation of Equation (8).

where $H_{FC} = X(X^\top Q^\top QX + \lambda I)^{-1} X^\top Q^\top Q$ and $H_{PC} = X(X^\top Q^\top QX + \lambda I)^{-1} X^\top$. So the degrees of freedom of $\widehat{\beta}_\alpha(\lambda)$ is $\mathrm{df} = \alpha_{PC} \operatorname{tr}(H_{PC}) + \alpha_{FC} \operatorname{tr}(H_{FC})$. In particular, both the fully and partially compressed estimators have simple forms for the degrees of freedom which do not need to be estimated.

For the linear combination estimator when α is estimated, computing the degrees of freedom is more complicated. This estimator is nonlinear because both the matrix $B(\lambda)$ and the weight vector $\widehat{\alpha}$ are functions of Y. A straighforward estimator of the degrees of freedom for $\widehat{\beta}_{\widehat{\alpha}}(\lambda)$ is created by replacing α with $\widehat{\alpha}$:

$$\widehat{\mathrm{df}} = \widehat{\alpha}_{FC} \operatorname{tr}\left(X(X^\top Q^\top QX + \lambda I)^{-1} X^\top Q^\top Q\right) + \widehat{\alpha}_{PC} \operatorname{tr}\left(X(X^\top Q^\top QX + \lambda I)^{-1} X^\top\right).$$

This approximation has been proposed for other nonlinear estimators such as neural networks (Ingrassia and Morlini, 2007). However, this estimator should intuitively underestimate the degrees of freedom, because it does not account for the extra flexibility introduced by allowing $\widehat{\alpha}$ to depend on Y.

Alternatively, the degrees of freedom can be computed via Stein's lemma (Stein, 1981) if we are willing to assume that the response vector is multivariate normal: $Y \sim N(\mu, \Sigma)$. Then, if $g(Y)$ is continuous and almost differentiable in Y, $\mathrm{df}(g) = \mathbb{E}[(\nabla \cdot g)(Y)]$, where $(\nabla \cdot g)(Y) = \sum_{i=1}^n \partial g_i / \partial Y_i$ is the *divergence* of g. It immediately follows that $\widehat{\mathrm{df}}(g) = (\nabla \cdot g)(Y)$ is an unbiased estimator of $\mathrm{df}(g)$. Though the calculus is tedious, the divergence of the linear combination estimator can be calculated by repeated applications of the chain rule. It turns out that the divergence is a perturbation of the approximate estimator $\widehat{\mathrm{df}}$ above. The explicit formula and its derivation are given in the Supplementary Material. We use the divergence for all of our empirical examples.

A major advantage of the divergence calculation is that it measures how hard the method fits the data through both $\widehat{\beta}$ and $\widehat{\alpha}$ for each value of λ. It thus allows the combination estimators to avoid overfitting through the choice of λ alone. As such, α is not a tuning parameter but rather a component of the estimator, and the only tuning parameter is λ. It is, however, not possible to jointly solve for the vector $(\beta,\ \alpha)$, because only their product is identified. Both combination estimators are essentially model-averaged estimators, and as such, their improved predictive performance in simulations is not surprising (see Section 4.5).[4] Their

[4]They are also better estimators of the true parameters (Section 4.2), which is not always the case for

benefit is that we can average in a data driven fashion without overfitting because of the ability to correctly incorporate its effect into the risk estimator.

3.2 Computing the path

In order to select tuning parameters, we need to compute the estimators quickly for a range of possible λ. Luckily, this can be implemented in the same way as with ridge regression. One can write $(X^\top Q^\top Q X + \lambda I)^{-1} = R(L^2 + \lambda I)^{-1} R^\top$, where the singular value decomposition is written $QX = SLR^\top$ with $S^\top S = I_q$, $R^\top R = I_p$, and L is a diagonal matrix of singular values. Therefore, we can take the SVD of QX once and then compute the entire path of solutions for a sequence of λ while only increasing the computational complexity multiplicatively in the number of λ values considered.

4 Simulations

We construct simulations under a variety of data generating scenarios to explore when $\widehat{\beta}_{\widehat{\alpha}}(\lambda)$ performs well. For the sake of space, we have only included a representative subset of the results. The remainder are available in the Supplementary Material. All simulations and empirical calculations were performed with `R` (R Core Team, 2019). Figures and tables are generated using the `tidyverse` family of packages (Wickham, 2017). The Supplemental Materials were created with `knitr` and `rmarkdown` (Xie, 2015, 2019; Xie et al., 2018). Most computations were implemented in parallel on the Carbonate[5] large memory computer cluster via the `batchtools` package (Lang et al., 2017).

4.1 Setup

To create data, we generate the design matrix $X \in \mathbb{R}^{n \times p}$ by independently sampling the rows from a multivariate normal distribution with mean zero and covariance matrix Σ which has unit variance on the diagonal and correlation $\rho \in \{0.1,\ 0.5,\ 0.9\}$ off the diagonal. We

model averaging.

[5]This research was supported in part by Lilly Endowment, Inc., through its support for the Indiana University Pervasive Technology Institute.

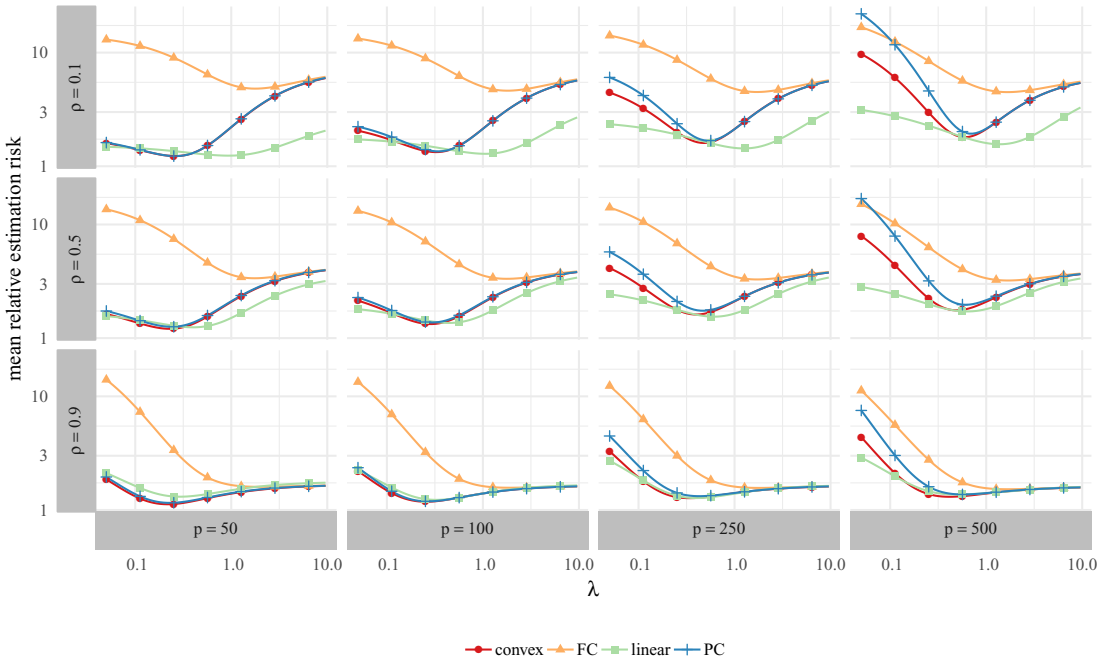

Figure 2: Average relative estimation risk compared to ridge regression (the Bayes-optimal estimator). Here β_* is Gaussian and $q = 1000$.

then form $Y = X\beta_* + \epsilon$, where ϵ_i are i.i.d. Gaussian with mean zero and variance σ^2. In all cases, we take $n = 10{,}000$ training samples and let $p = 50,\ 100,\ 250$, or 500.

We use two different structures for β_*, though we have only included results for the first here. In the first case, we take $\beta_* \sim \mathrm{N}(0, \tau^2 I)$. Under this model, ridge regression is Bayes-optimal for $\lambda_* = n^{-1}\sigma^2/\tau^2$, so it represents a true basis for comparing the loss in prediction accuracy due to compression. We set $\tau^2 = \pi/2$ so that $p^{-1}\mathbb{E}[||\beta_*||_1] = 1$. Finally, to ensure that λ_* is not too small, we take $\sigma = 50$ implying $\lambda_* \approx 0.16$. The second structure for β_* is created to make ridge regression perform poorly: we simply set $(\beta_*)_j \equiv 1$ for all $1 \leq j \leq p$.

We examine four different compressed estimators with penalization: (1) full compression, (2) partial compression, (3) a linear combination of the first two, and (4) a convex combination of the first two. We also use the OLS estimator and the ridge regression estimator. For ridge regression, we use λ_* in the first scenario and choose λ by minimizing GCV in the other case. For the compressed estimators, we examine three possible values of $q \in \{500,\ 1000,\ 2000\}$. In each case, we generate $Q \in \{-1, 0, 1\}^{q \times n}$ as a "sparse Bernoulli" matrix with $s = 3$.

4.2 Estimation error simulations

For all four compressed methods, there exist λ values which allow the compressed method to beat ordinary least squares (not shown). This is to be expected given the simulation conditions. Analogously, as expected, regularized compression always outperforms unregularized compression for some values of λ. While we have simulated all combinations of q, p, and ρ, we only display results for $q = 1000$ which are similar to those from other parameter configurations. Figure 2 shows the estimation risk for the compressed methods relative to the estimation risk for ridge regression when β_* is drawn from a Gaussian distribution.

Combining partial and full compression strictly dominates the other compressed methods in terms of estimation risk at all values of λ. When p is small relative to n and the design has high correlation, tuning parameter selection is less important for accurate estimation. We note also that regardless of the choice of design, the linear combination is generally less sensitive to the choice of λ.

4.3 Selecting tuning parameters

The previous simulation shows that regularized compression can compare favorably with the performance of the Bayes estimator as long as we can choose λ well. In this section, we investigate our proposed tuning parameter selection strategy.

We again generate a training set with $n = 10{,}000$, but we also generate an independent test set with 5000 samples. Then we use the training set to choose λ_{GCV} by generalized cross validation as described in Section 3.1. We also define an optimal (though unavailable) λ_{test} by minimizing the test set prediction error,

$$\lambda_{\text{test}} = \underset{\lambda}{\text{argmin}} \frac{1}{n} \left|\left| Y_{\text{test}} - X_{\text{test}} \widehat{\beta}(\lambda) \right|\right|_2^2.$$

Figure 3 shows the prediction risk of the GCV-selected estimates relative to the oracle for $q = 1000$ and $\rho = 0.5$, but results for other choices of q and ρ are similar. That is, we plot the ratio

$$\frac{\text{test}_{\text{GCV}}}{\text{test}_{\text{min}}} = \frac{\left|\left| Y_{\text{test}} - X_{\text{test}} \widehat{\beta}(\lambda_{\text{GCV}}) \right|\right|_2^2}{\left|\left| Y_{\text{test}} - X_{\text{test}} \widehat{\beta}(\lambda_{\text{test}}) \right|\right|_2^2} \geq 1.$$

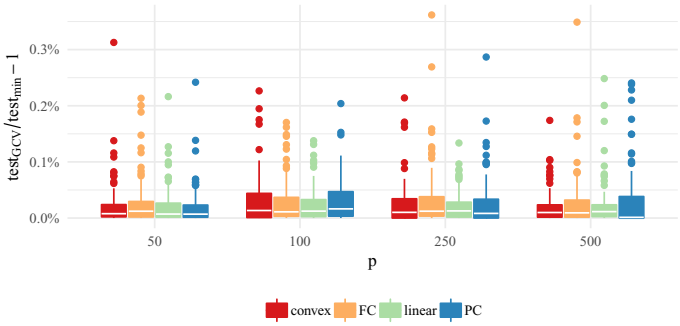

Figure 3: Percentage increase in test error between the tuning parameter chosen by GCV and the best tuning parameter we would have chosen with access to a large test set (necessarily greater than 0).

All methods are within 1% of the best test error the majority of the time, but the full compression tends to be the worst. Note that the minimum test error is for the particular method rather than relative to the best possible test error across all methods. Thus, while GCV selects the optimal tuning parameter for partial compression more accurately than for the linear combination, the linear combination has lower estimation error at its own GCV-selected tuning parameter. While Section 3 presented two methods for estimating the degrees of freedom—a simple plug-in approximation and the divergence—we only present the results for the divergence.

4.4 Overall performance assessments

Using the tuning parameter selected by GCV with the divergence, different compression methods perform better in different situations. Figure 4 examines the performance across all simulations when β_* is Gaussian. Specifically, for each of the 50 training data sets, we estimate the linear model with each method, choosing λ by GCV if appropriate. We select the method with the lowest estimation error. We plot the proportion of times each method "wins" across all simulation conditions. Overall, the convex combination estimator works well in most cases and has smaller variance than the linear combination estimator. In the cases in which partial compression is better, the convex combination is close behind. The performance of the linear combination estimator is somewhat odd. Because the relative weights on full compression and partial compression are data dependent and unconstrained,

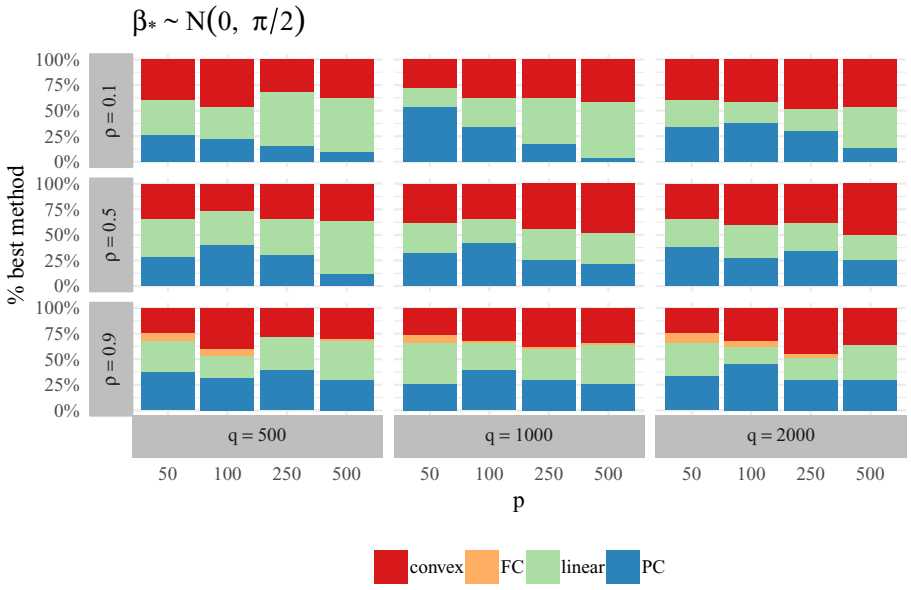

Figure 4: Proportion of simulations a method has the best estimation error for each simulation condition.

these weights can explode despite the regularization imposed by λ. Thus, it may be that $\widehat{\beta}_{FC} \approx 0$ and $\widehat{\beta}_{PC} \approx 0$, but by multiplying them by large magnitude $\widehat{\alpha}_{FC}$ and $\widehat{\alpha}_{PC}$ of opposite signs, the result is a large unpenalized estimate. In cases where the true coefficients are similar to each other but non-zero, the effect is to shrink toward that non-zero value. In these situations, the linear combination estimator works well, while in others, it's behavior becomes increasingly erratic as $\lambda \to \infty$. Based on the simulations, we feel that the convex combination estimator is the best choice in most situations.

4.5 Recommendations

Even though ridge regression is optimal when β_* has a Gaussian distribution, regularized compression can achieve nearly the same prediction error while using fewer computations and less storage space. Figure 5 displays the prediction risk for each method when the tuning parameter is chosen by GCV. The difference between the optimal model and the compressed approximations is often less than 5%. Compressed methods work well when the correlation in the design is high, which is rather unexpected, or when either p/n is small or $q/n \to 1$.

Under the setting $\beta_* \equiv 1$, partial compression works relatively poorly, and therefore,

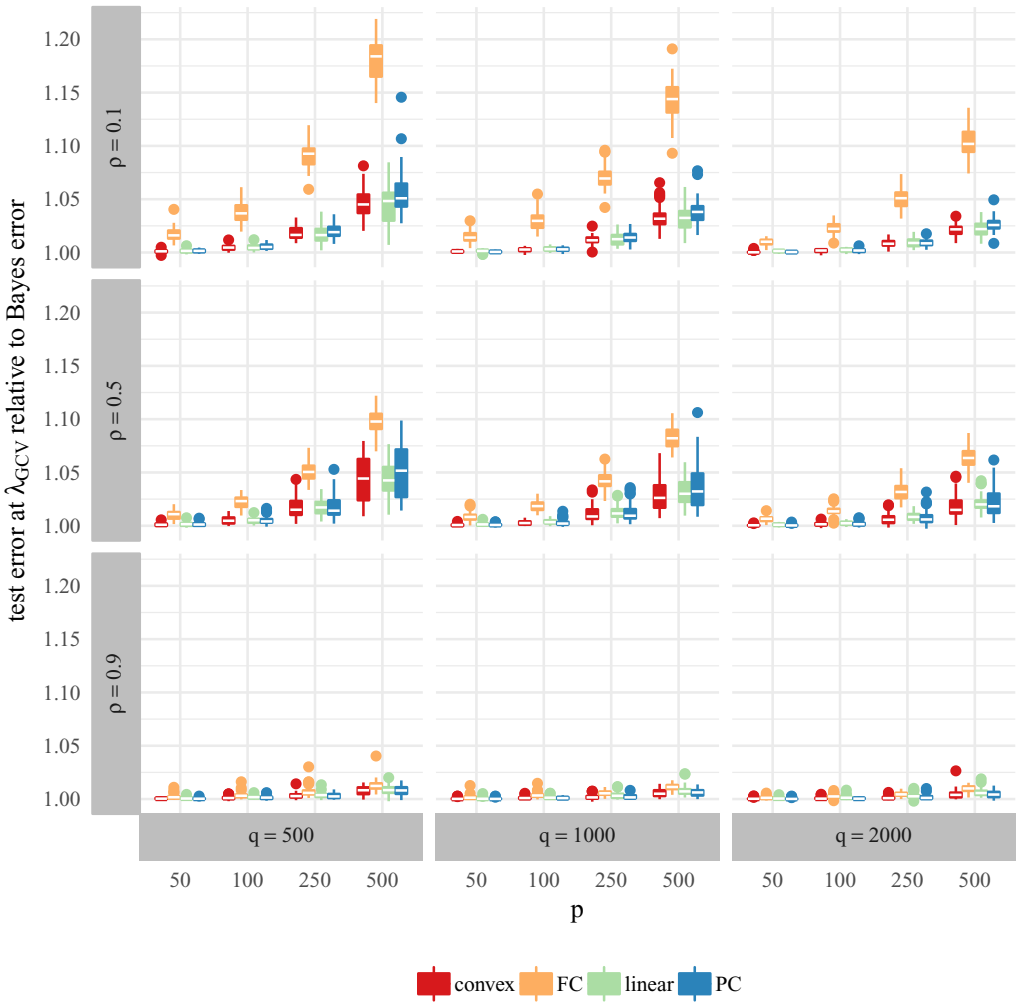

Figure 5: Prediction error on the test set for all methods.

$\widehat{\alpha}$ tends to emphasize full compression. As discussed above, partial compression is biased toward 0 even when $\lambda = 0$, and larger λ will exacerbate this property. In this case, both convex combination and linear combination work significantly better. The complete set of simulation results are available in the Supplementary Material.

5 Real data examples

We present results for two different types of real data: a collection of genetics data and an exercise in denoising a magnetic resonance image (MRI) of the brain. While our simulations used perhaps idealized conditions (true linear model, homoskedastic noise), our data exam-

ples are chosen to illustrate performance under more varied conditions. The first maintains independence across observations and homoskedasticity but uses entirely categorical predictors. The second uses continuous predictors but with heteroskedastic, spatially dependent noise.

5.1 Genetics

The first data we examine are a collection of short-read RNA sequences. The data are publicly available[6] and were first examined by Li et al. (2010), who suggest a Poisson linear model to predict read counts based on the surrounding nucleotides. An implementation of this method is provided in the `R` package `mseq`, described by Li et al. (2010) and available from the CRAN archive.

In all, there are eight data files from three research groups. Three datasets are due to Mortazavi et al. (2008) which mapped mouse transcriptomes from brain, liver, and skeletal muscle tissues. Wang et al. (2008) collected data from 15 different human tissues which have been merged into three groups based on tissue similarities. Finally, Cloonan et al. (2008) examined RNA sequences from mouse embryonic stem cells and embryoid bodies. In all cases, we use the top 100 highly-expressed genes as well as the surrounding sequences to predict expression counts as in Li et al. (2010). Table 1 shows the sample size n for each of the eight data sets.

Following Li et al. (2010) and Dalpiaz et al. (2013), we examine each of these datasets separately. In order to build the model, we must select how many surrounding nucleotides to use for prediction. As nucleotides are factors (taking levels C,T,A,G), a window of k surrounding nucleotides will give $p = 3(k+1)$ binary predictors plus an intercept. We could also use dinucleotide pairs (or higher interactions), as in Li et al. (2010), resulting in $p = 15(k+1)$ predictors. For our illustration, we follow Ma et al. (2015), who also apply different compressed linear regression methods to these data, and use $k = 39$.

The results of our analysis are shown in Figure 6. For each dataset, which we denote by the first letter of the senior author's last name (W, B, and G respectively) followed by a number, we split the data randomly into 75% training data and 25% testing data.

[6] From Jun Li: `http://www3.nd.edu/~jli9/mseq/data_top100.zip`

dataset	B1	B2	B3	G1	G2	W1	W2	W3
n	157614	125056	103394	51751	64966	146828	171776	143570

Table 1: Number of observations for each of the 8 genetics data sets.

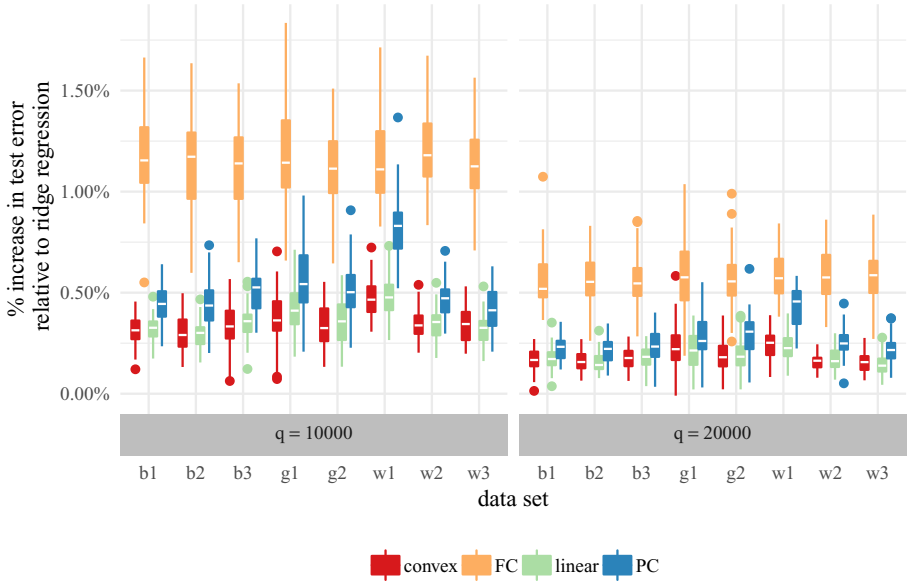

Figure 6: Results of each method on 50 replications of training test splits on each of the eight genetics data sets. The results are percentage increase in test error relative to ordinary least squares.

We then compress the training set using $q = 10{,}000$ and $q = 20{,}000$. We apply each of the regularized compressed methods, choosing λ by generalized cross validation and then evaluate the estimators by making predictions on the test set. We repeat this procedure 50 times and present the average of the log test error relative to ridge regression. Across data sets, $q = 10{,}000$ results in data reductions between 74% and 93% (meaning $0.26 \geq q/n \geq 0.07$) while $q = 20{,}000$ gives reductions between 48% and 84%.

For these data, ridge and OLS give equivalent test set performance (differing by less than .001%) across all data sets. We also tried LASSO, adaptive LASSO, and the elastic net which are all nearly equivalent to ridge regression.[7] Previous analyses have found that

[7]The full set of results are displayed in the Supplementary Material. Computations were performed with the `glmnet` package (Friedman et al., 2010; Simon et al., 2011)

the coefficients are roughly centered around zero with most quite small. Nonetheless, the compressed methods are not much worse than ridge regression. The worst method is always full compression. The linear combination and the convex combination are nearly equivalent, while partial compression is just slightly worse. Even for full compression, its worst performance across all data sets and over both values of q is less than 1.5% worse than OLS. So even for the worst performing method, large amounts of compression result in a negligible increase in test set error.

5.2 MRI denoising

A crucial preprocessing step for medical image analysis is the removal of spurious noise. Pre-analysis denoising is especially crucial for the study of Magnetic Resonance Images (MRIs) because the aim is detection or visualization of local structures (brain lesions, tumors, white matter fiber tracking) rather than voxel-level (the 3D image pixel) deviations. Standard methods use variations of kernel smoothing (Saint-Marc et al., 1989) or total variation denoising (Sapiro, 1996), but more recent methods attempt to adapt to edge or other structures automatically to avoid smoothing neighboring regions together (Coupe et al., 2008). These current methods, called nonlocal means denoising filters, come at a computational cost: a typical 3D T1-weighted MRI has around 7 million voxels and these may be scanned repeatedly over time.

In this section, we follow the analysis of Coupe et al. (2008) for denoising a single $181 \times 217 \times 181$ 3D MRI. Our goal is not to improve upon state-of-the-art MR denoising methods, but simply to illustrate that reasonable compressed approximations can perform adequately in a small fraction of the time. In particular, we will simply use convex combination compression to regress individual voxels on a 3D neighborhood. As such, our linear model will have a strongly correlated design matrix X as well as correlated, heteroskedastic noise. Coupe et al. (2008) find that standard nonlinear means desnoising requires around 6 hours of processing time on a single 3 GHz CPU while their optimized approximation takes about 5 minutes with negligible loss of performance. Performing (admittedly non-optimized) convex combination regularized regression requires between 30 seconds and 3 minutes depending on the size of the local neighborhood and the amount of compression used.

As in Coupe et al. (2008), we use a simulated, noise-free, T1-weighted MRI from the

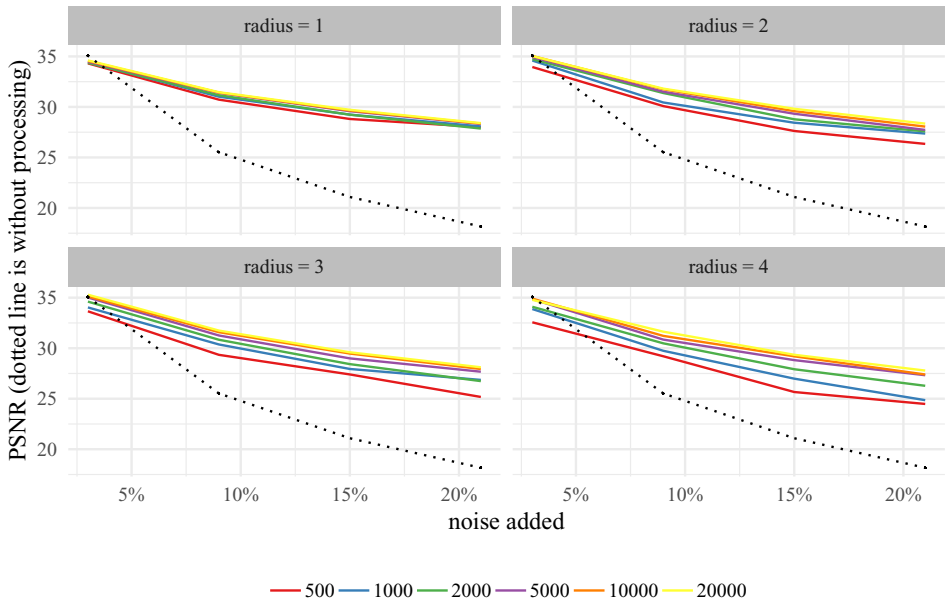

Figure 7: Prediction accuracy of convex compression for a variety of noise levels and compression sizes. Larger values of the PSNR statistic are better.

BrainWeb database (Collins et al., 1998). We follow their analysis and simulate noisy images by adding independent Gaussian noise to each voxel. We create a mask of voxels to use by keeping only those voxels whose true brightness exceeds 50, treating others as non-brain. We use the bottom 130 slices of the noisy data as our training set and the top 40 slices as a validation set for choosing the amount of regularization. Finally, we evaluate our predictions on the original, noise-free brain.

Figure 7 displays our results for six different compression levels—$q \in \{500, 1000, 2000, 5000, 10000, 20000\}$—4 different noise levels—3%, 9%, 15%, and 21%— and 4 different radii for the local neighborhood (meaning that the predictors are the noisy values in a 3D cube of radius r around the target voxel). The noise standard deviation is taken to be 150 times the noise percent. In each case, the training data has n = 2,651,260, the validation set has n = 231,340 and the test set has just over 3 million voxels. The number of predictors $p = \{26,\ 124,\ 342,\ 728\}$ respectively. The measure of accuracy we use is the peak signal-to-noise ratio, defined for 8-bit encoded images as $\text{PSNR} = 20 \log_{10} \left(255 / \sqrt{\frac{1}{n} \sum_i (Y_i - \widehat{Y}_i)^2} \right)$. The PSNR (measured in dB) is decreasing in the mean-squared error, so larger values are better. As shown in the figure, at the 3% noise level, this simple regression technique is

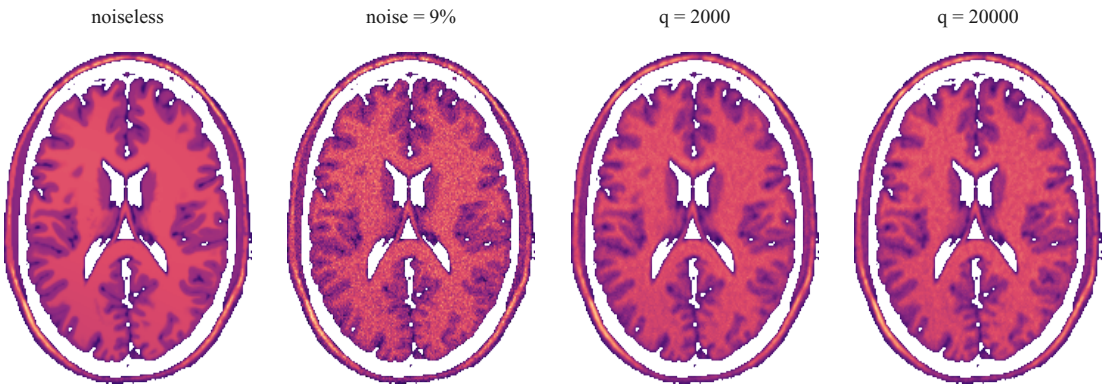

Figure 8: This image shows the performance of our methods on one slice. Our best result at this noise level used $q = 20000$ and radius 2 and achieved a PSNR of 31.8dB. For comparison, state-of-the-art approximations have PSNR between 33.7dB and 34.4dB. The $q = 2000$ image has a PSNR of 31.4dB while the noisy image has PSNR of 25.5dB.

unable to improve on simply using the noisy image no matter the size of the local radius or the amount of compression. However, at larger noise levels, this method performs reasonably. For comparison to the state-of-the-art technique, at the 9% noise level, Coupe et al. (2008, see Table II) get PSNR = 34.44dB using their best method which requires 50 minutes to compute and PSNR = 33.75dB using their fastest method (requiring 5.5 minutes). Our best result at this noise level was PSNR = 31.8dB, used a radius of 2, q = 20,000, and required 27 seconds to compute. Figure 8 gives a visual comparison of these results for the middle horizontal slice. The noisy image had PSNR = 25.5dB.

6 Theoretical analysis

To develop a better understanding of the relationship between the compressed regression methods proposed here and standard full-data techniques, we derive expressions for the expectation and variance of full and partial compression estimators as well as their bias and variance. For comparison, we first present the standard analogues for ridge regression.

6.1 Mean squared error performance

Write the singular value decomposition of $X = UDV^\top$ where $U^\top U = I_n$, $V^\top V = I_p$, and D is a diagonal matrix of singular values. Then define the ridge regression estimator of β_* as in Equation (6).

Standard analysis gives the squared bias and trace of the variance of the ridge regression estimator conditional on the design.

Lemma 1.

$$\text{bias}^2\left(\widehat{\beta}_{ridge}(\lambda) \mid X\right) = \lambda^2 \beta_*^\top V(D^2 + \lambda I_p)^{-2} V^\top \beta_*.$$
$$\text{tr}\left(\mathbb{V}[\widehat{\beta}_{ridge}(\lambda) \mid X]\right) = \sigma^2 \sum_{j=1}^{p} \frac{d_j^2}{(d_j^2 + \lambda)^2}.$$

In what follows, we will derive approximations to these quantities for the fully and partially compressed ridge regression estimators of β_*. Because all of our estimators depend on $(X^\top Q^\top Q X + \lambda I_p)^{-1}$, a generally intractable quantity, we derive approximate results via a first order Taylor expansion of the estimator with respect to the matrix $Q^\top Q$. The proofs as well as intermediary results are included in the Supplementary Material.

Following Ma et al. (2015), we use the Taylor expansion of $\widehat{\beta}$ as a function of $A := \frac{s}{q} Q^\top Q$ around I_n to derive results conditional on Y and X (taking expectations over Q) as well as results unconditional on Y. The first case reflects the randomness in the compression algorithm relative to the more computationally demanding ridge regression. The second is useful for comparing the compressed procedures with ridge regression by including randomness introduced through the data generating process and through the compression algorithm. In all cases, these results are conditional on the design matrix as was the case above. For convenience of expression, define

$$M := (X^\top X + \lambda I_p)^{-1} X^\top = V(D^2 + \lambda I_p)^{-1} D U^\top, \text{ and}$$
$$H := X(X^\top X + \lambda I_p)^{-1} X^\top = UD(D^2 + \lambda I_p)^{-1} D U^\top = XM.$$

Finally, for these results we will assume the linear model with homoskedastic noise and that $q = cn$ for some $0 < c \leq 1$ which is fixed. We discuss this last assumption further in the remark below.

Theorem 2. *The squared-bias of the fully compressed estimator is:*

$$\begin{aligned}\operatorname{bias}^2\left(\widehat{\beta}_{FC} \mid X, Q\right) &= \lambda^2 \beta_*^\top V (D^2 + \lambda I)^{-2} V^\top \beta_* + o_P(1) + \\ &\qquad 2\beta_*^\top (MX - I)^\top M (A - I)(I - H) X \beta_* \\ \operatorname{bias}^2\left(\widehat{\beta}_{FC} \mid X\right) &= \lambda^2 \beta_*^\top V (D^2 + \lambda I)^{-2} V^\top \beta_* + o_P(1).\end{aligned}$$

The squared-bias of the partially compressed estimator is:

$$\begin{aligned}\operatorname{bias}^2\left(\widehat{\beta}_{PC} \mid X, Q\right) &= \lambda^2 \beta_*^\top V (D^2 + \lambda I)^{-2} V^\top \beta_* - 2\beta_*^\top (MX - I)^\top M (A - I) H X \beta_* + o_P(1) \\ \operatorname{bias}^2\left(\widehat{\beta}_{PC} \mid X\right) &= \lambda^2 \beta_*^\top V (D^2 + \lambda I)^{-2} V^\top \beta_* + o_P(1).\end{aligned}$$

Note that, ignoring the remainder, the squared bias of both fully compressed and partially compressed estimators when averaged over the compression is the same as that of ridge regression.

Theorem 3. *The variance of the fully compressed estimator is:*

$$\begin{aligned}\operatorname{tr}\left(\mathbb{V}\left[\widehat{\beta}_{FC} \mid X, Q\right]\right) &= \sigma^2 \sum_{j=1}^{p} \frac{d_j^2}{(d_j^2 + \lambda)^2} + 2 \operatorname{tr}\left(M(I - H)(A - I)M^\top\right) + o_P(1) \\ \operatorname{tr}\left(\mathbb{V}\left[\widehat{\beta}_{FC} \mid X\right]\right) &= \sigma^2 \sum_{j=1}^{p} \frac{d_j^2}{(d_j^2 + \lambda)^2} + o_P(1) \\ &\quad + \frac{\lambda^2 (s-2)_+}{q} \beta_*^\top V D^2 (D^2 + \lambda I)^{-4} D^2 V^\top \beta_* \\ &\quad + \frac{\lambda^2}{q} \beta_*^\top V D (D^2 + \lambda I)^{-2} D V^\top \beta_* \sum_{j=1}^{p} \frac{d_j^2}{(d_j^2 + \lambda)^2}.\end{aligned}$$

The variance of the partially compressed estimator is:

$$\begin{aligned}\operatorname{tr}\left(\mathbb{V}\left[\widehat{\beta}_{PC} \mid X, Q\right]\right) &= \sigma^2 \sum_{j=1}^{p} \frac{d_j^2}{(d_j^2 + \lambda)^2} + 2 \operatorname{tr}\left(MH(A - I)M^\top\right) + o_P(1) \\ \operatorname{tr}\left(\mathbb{V}\left[\widehat{\beta}_{PC} \mid X\right]\right) &= \sigma^2 \sum_{j=1}^{p} \frac{d_j^2}{(d_j^2 + \lambda)^2} + o_P(1) \\ &\quad + \frac{(s-2)_+}{q} \beta_*^\top V D^2 (D^2 + \lambda I)^{-2} D^2 V^\top \beta_* \\ &\quad + \frac{1}{q} \beta_*^\top V D^3 (D^2 + \lambda I)^{-2} D^3 V^\top \beta_* \sum_{j=1}^{p} \frac{d_j^2}{(d_j^2 + \lambda)^2}.\end{aligned}$$

Remark 1. *In each expression above, the Taylor series is valid whenever higher-order terms are small. In our case, the higher-order terms are* $o_P(||A-I||^2)$ *under the expansion, for some matrix norm* $||\cdot||$*. Here* $o_P(\cdot)$ *is with respect to the randomness in* A *through* Q*. As* Q *is independent of the data, one can examine how large these deviations are likely to be. In particular, using results similar to the Tracy-Widom law (Rudelson and Vershynin, 2010, Proposition 2.4), one can show that* $||A-I|| = O_P(\sqrt{n/q})$*. Therefore, taking* $q = cn$ *for some* $0 < c \leq 1$ *means that the remainder is* $o_P(1)$*. This is in contrast with results of Ma et al. (2015) for two reasons: (1) their sampling mechanism is allowed to depend on the data where ours is not, and (2) they can lose rank from the compression. In our case,* $(X^\top Q^\top QX + \lambda I)$ *is full rank for all* $\lambda > 0$ *regardless of the rank of* $X^\top Q^\top QX$*, so the Taylor series is always valid.*

Corollary 4. *Suppose* X *is such that* $\frac{1}{n}X^\top X = I_p$*,* $b^2 := ||\beta_*||_2^2$*, and* $\theta := \lambda/n$*. Then*

$$MSE(\widehat{\beta}_{ridge}) = b^2\left(\frac{\theta}{1+\theta}\right)^2 + \frac{p\sigma^2}{n(1+\theta)^2}$$

$$MSE(\widehat{\beta}_{FC}) = b^2\left(\frac{\theta}{1+\theta}\right)^2 + \frac{p\sigma^2}{n(1+\theta)^2} + \frac{b^2p\theta^2(s-2)_+}{q(1+\theta)^4} + \frac{p^2\theta^2b^2}{q(1+\theta)^4}$$

$$MSE(\widehat{\beta}_{PC}) = b^2\left(\frac{\theta}{1+\theta}\right)^2 + \frac{p\sigma^2}{n(1+\theta)^2} + \frac{p(s-2)_+b^2}{q(1+\theta)^2} + \frac{pb^2}{q(1+\theta)^4}.$$

Hence, for ridge, the optimal $\theta_* = \sigma^2p/(nb^2)$ and $\lambda_* = \sigma^2p/b^2$. For the other methods, the MSE can be minimized numerically, but we have, so far, been unable to find an analytic expression.

7 Conclusion

In this paper, we propose and explore a broad family of compressed, regularized linear model estimators created by generalizing two commonly used approximation procedures which we notated $\widehat{\beta}_{FC}$ and $\widehat{\beta}_{PC}$. We show that $\widehat{\beta}_{FC}$ must indeed perform worse than the least squares solution. We suggest combining full and partial compression estimators instead with an ℓ_2 penalty. As this additional regularization introduces a tuning parameter, we give justifiable methods for choosing it in a computationally feasible way. We find that our new estimators can perform nearly as well as their full-data analogues, and that our tuning parameter selection methods are quite accurate.

Interesting future work would examine other forms for the compression matrix Q to examine their statistical impact. Finally, while ridge penalties are amenable to theoretical analysis because of their closed form solution, it is worthwhile to examine other penalty functions and other losses, such as generalized linear models.

SUPPLEMENTARY MATERIAL

Complete simulation results, divergence formula, proofs: Figures for additional simulation conditions not included above, the explicit formula for the divergence used in tuning parameter selection and proofs of all the results contained in Section 6. (PDF document)

R-package "cplr": R-package containing code to perform the methods described in the article. The package also contains all data sets used as examples in the article. (GNU zipped tar) The package is also easily installable by calling `devtools::install_github('dajmcdon/cplr')` from R.

References

Achlioptas, D. (2003), 'Database-friendly random projections: Johnson-Lindenstrauss with binary coins', *Journal of Computer and System Sciences* **66**(4), 671–687.

Ailon, N. and Chazelle, B. (2006), Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform, *in* 'Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing', ACM, pp. 557–563.

Avron, H., Maymounkov, P. and Toledo, S. (2010), 'Blendenpik: Supercharging LAPACK's least-squares solver', *SIAM Journal on Scientific Computing* **32**(3), 1217–1236.

Bair, E., Hastie, T., Paul, D. and Tibshirani, R. (2006), 'Prediction by supervised principal components', *Journal of the American Statistical Association* **101**(473), 119–137.

Becker, S., Kawas, B., Petrik, M. and Ramamurthy, K. N. (2017), Robust partially-compressed least-squares, *in* 'The Thirty-First AAAI Conference on Artificial Intelligence'.

Cloonan, N., Forrest, A. R. R., Kolle, G., Gardiner, B. B. A., Faulkner, G. J., Brown, M. K., Taylor, D. F., Steptoe, A. L., Wani, S., Bethel, G., Robertson, A. J., Perkins, A. C., Bruce, S. J., Lee, C. C., Ranade, S. S., Peckham, H. E., Manning, J. M., McKernan, K. J. and Grimmond, S. M. (2008), 'Stem cell transcriptome profiling via massive-scale mRNA sequencing', *Nature Methods* **5**(7), 613–619.

Collins, D. L., Zijdenbos, A. P., Kollokian, V., Sled, J. G., Kabani, N. J., Holmes, C. J. and Evans, A. C. (1998), 'Design and construction of a realistic digital brain phantom', *IEEE Transactions on Medical Imaging* **17**(3), 463–468.

Coupe, P., Yger, P., Prima, S., Hellier, P., Kervrann, C. and Barillot, C. (2008), 'An optimized blockwise nonlocal means denoising filter for 3-D magnetic resonance images', *IEEE Transactions on Medical Imaging* **27**(4), 425–441.

Dalpiaz, D., He, X. and Ma, P. (2013), 'Bias correction in RNA-Seq short-read counts using penalized regression', *Statistics in Biosciences* **5**(1), 88–99.

Dasgupta, A., Kumar, R. and Sarlós, T. (2010), A sparse Johnson-Lindenstrauss transform, *in* 'Proceedings of the 42nd ACM Symposium on Theory of Computing', ACM, pp. 341–350.

Ding, L. and McDonald, D. J. (2017), 'Predicting phenotypes from microarrays using amplified, initially marginal, eigenvector regression', *Bioinformatics* **33**(14), i350–i358.

Drineas, P., Magdon-Ismail, M., Mahoney, M. W. and Woodruff, D. P. (2012), 'Fast approximation of matrix coherence and statistical leverage', *Journal of Machine Learning Research* **13**(Dec), 3475–3506.

Drineas, P., Mahoney, M. W., Muthukrishnan, S. and Sarlós, T. (2011), 'Faster least squares approximation', *Numerische Mathematik* **117**(2), 219–249.

Efron, B. (1986), 'How biased is the apparent error rate of a prediction rule?', *Journal of the American Statistical Association* **81**(394), 461–470.

Frey, R. A., Ackerman, S. and Soden, B. J. (1996), 'Climate parameters from satellite

spectral measurements. Part 1: Collocated AVHRR and HIRS/2 observations of spectral greenhouse parameter', *Journal of Climate* **9**(2), 327–344.

Friedman, J., Hastie, T. and Tibshirani, R. (2010), 'Regularization paths for generalized linear models via coordinate descent', *Journal of Statistical Software* **33**(1), 1–22.

Gittens, A. and Mahoney, M. (2013), Revisiting the Nystrom method for improved large-scale machine learning, *in* S. Dasgupta and D. McAllester, eds, 'Proceedings of the 30th International Conference on Machine Learning (ICML-13)', Vol. 28, JMLR Workshop and Conference Proceedings, pp. 567–575.

Golub, G. H., Heath, M. and Wahba, G. (1979), 'Generalized cross-validation as a method for choosing a good ridge parameter', *Technometrics* **21**(2), 215–223.

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

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

Hoerl, A. E. and Kennard, R. W. (1970), 'Ridge regression: Biased estimation for nonorthogonal problems', *Technometrics* **12**(1), 55–67.

Homrighausen, D. and McDonald, D. J. (2016), 'On the Nyström and column-sampling methods for the approximate principal components analysis of large data sets', *Journal of Computational and Graphical Statistics* **25**(2), 344–362.

Ingrassia, S. and Morlini, I. (2007), *Equivalent Number of Degrees of Freedom for Neural Networks*, Advances in Data Analysis, Springer Berlin Heidelberg, Berlin, Heidelberg.

Kane, D. M. and Nelson, J. (2014), 'Sparser Johnson-Lindenstrauss transforms', *Journal of the ACM* **61**(1), Article 4.

Lang, M., Bischl, B. and Surmann, D. (2017), 'batchtools: Tools for R to work on batch systems', *The Journal of Open Source Software* **2**(10).

Li, J., Jiang, H. and Wong, W. H. (2010), 'Modeling non-uniformity in short-read rates in RNA-Seq data', *Genome Biology* **11**(5), 1–11.

Ma, P., Mahoney, M. W. and Yu, B. (2015), 'A statistical perspective on algorithmic leveraging', *The Journal of Machine Learning Research* **16**(1), 861–911.

Mallows, C. L. (1973), 'Some comments on C_p', *Technometrics* **15**(4), 661–675.

Meng, X., Saunders, M. A. and Mahoney, M. W. (2014), 'LSRN: A parallel iterative solver for strongly over- or under-determined systems', *SIAM Journal on Scientific Computing* **36**(2), C95–C118.

Mortazavi, A., Williams, B. A., McCue, K., Schaeffer, L. and Wold, B. (2008), 'Mapping and quantifying mammalian transcriptomes by RNA-Seq', *Nature Methods* **5**(7), 621–628.

Paul, D., Bair, E., Hastie, T. and Tibshirani, R. (2008), "Preconditioning' for feature selection and regression in high-dimensional problems', *The Annals of Statistics* **36**(4), 1595–1618.

Pilanci, M. and Wainwright, M. J. (2015), 'Iterative Hessian sketch: Fast and accurate solution approximation for constrained least-squares', *Journal of Machine Learning Research* pp. 1–33.

R Core Team (2019), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.
URL: *https://www.R-project.org/*

Raskutti, G. and Mahoney, M. (2015), Statistical and algorithmic perspectives on randomized sketching for ordinary least-squares, *in* F. Bach and D. Blei, eds, 'Proceedings Proceedings of the 32nd International Conference on Machine Learning (ICML)', Vol. 37, PMLR, Lille, France, pp. 617–625.

Rokhlin, V. and Tygert, M. (2008), 'A fast randomized algorithm for overdetermined linear least-squares regression', *Proceedings of the National Academy of Sciences* **105**(36), 13212–13217.

Rudelson, M. and Vershynin, R. (2010), Non-asymptotic theory of random matrices: Extreme singular values, *in* R. Bhatia, A. Pal, G. Rangarajan, V. Srinivas and M. Vanninathan, eds, 'Proceedings of the International Congress of Mathematicians 2010 (ICM 2010)', pp. 1576–1602.

Saint-Marc, P., Chen, J.-S. and Medioni, G. (1989), Adaptive smoothing: A general tool for early vision, *in* 'Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition', IEEE, pp. 618–624.

Sapiro, G. (1996), From active contours to anisotropic diffusion: Connections between basic pde's in image processing, *in* 'Proceedings of the International Conference on Image Processing', Vol. 1, IEEE, pp. 477–480.

Simon, N., Friedman, J., Hastie, T. and Tibshirani, R. (2011), 'Regularization paths for Cox's proportional hazards model via coordinate descent', *Journal of Statistical Software* **39**(5), 1–13.

Staten, P. W., Kahn, B. H., Schreier, M. M. and Heidinger, A. K. (2016), 'Subpixel characterization of HIRS spectral radiances using cloud properties from AVHRR', *Journal of Atmospheric and Oceanic Technology* **33**(7), 1519–1538.

Stein, C. M. (1981), 'Estimation of the mean of a multivariate normal distribution', *The Annals of Statistics* **9**(6), 1135–1151.

Wang, E. T., Sandberg, R., Luo, S., Khrebtukova, I., Zhang, L., Mayr, C., Kingsmore, S. F., Schroth, G. P. and Burge, C. B. (2008), 'Alternative isoform regulation in human tissue transcriptomes', *Nature* **456**(7221), 470–476.

Wang, J., Lee, J., Mahdavi, M., Kolar, M. and Srebro, N. (2017), Sketching Meets Random Projection in the Dual: A Provable Recovery Algorithm for Big and High-dimensional Data, *in* A. Singh and J. Zhu, eds, 'Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)', Vol. 54 of *Proceedings of Machine Learning Research*, PMLR, Fort Lauderdale, FL, USA, pp. 1150–1158.

Wickham, H. (2017), 'tidyverse: Easily install and load the 'tidyverse'', *R package version 1.2.1* .
URL: *https://www.tidyverse.org*

Woodruff, D. P. (2014), 'Sketching as a tool for numerical linear algebra', *Foundations and Trends® in Theoretical Computer Science* **10**(1–2), 1–157.

Xie, Y. (2015), *Dynamic Documents with R and knitr*, 2nd edn, Chapman and Hall/CRC, Boca Raton, Florida.

Xie, Y. (2019), 'knitr: A general-purpose package for dynamic report generation in R', *R package version 1.22* .
URL: *https://yihui.name/knitr/*

Xie, Y., Allaire, J. and Grolemund, G. (2018), *R Markdown: The Definitive Guide*, Chapman and Hall/CRC, Boca Raton, Florida.

Zhang, L., Mahdavi, M., Jin, R., Yang, T. and Zhu, S. (2013), Recovering the optimal solution by dual random projection, *in* S. Shalev-Shwartz and I. Steinwart, eds, 'Proceedings of the 26th Annual Conference on Learning Theory', Vol. 30 of *Proceedings of Machine Learning Research*, PMLR, pp. 135–157.

Zhou, S., Lafferty, J. and Wasserman, L. (2009), 'Compressed and privacy-sensitive sparse regression', *IEEE Transactions on Information Theory* **55**(2), 846–866.