The Journal of Supercomputing
https://doi.org/10.1007/s11227-023-05309-w

Sketching the Krylov subspace: faster computation of the entire ridge regularization path

Yifei Wang[1] · Mert Pilanci[1]

Accepted: 14 April 2023

Abstract
We propose a fast algorithm for computing the entire ridge regression regularization path in nearly linear time. Our method constructs a basis on which the solution of ridge regression can be computed instantly for any value of the regularization parameter. Consequently, linear models can be tuned via cross-validation or other risk estimation strategies with substantially better efficiency. The algorithm is based on iteratively sketching the Krylov subspace with a binomial decomposition over the regularization path. We provide a convergence analysis with various sketching matrices and show that it improves the state-of-the-art computational complexity. We also provide a technique to adaptively estimate the sketching dimension. This algorithm works for both the over-determined and under-determined problems. We also provide an extension for matrix-valued ridge regression. The numerical results on real medium and large-scale ridge regression tasks illustrate the effectiveness of the proposed method compared to standard baselines which require super-linear computational time.

Keywords Ridge regression · Randomized algorithms · Kernel ridge regression

1 Introduction

We consider the following ridge regression problem

$$\min_{x\in\mathbb{R}^d} f(x) = \frac{1}{2}\|Ax-b\|_2^2 + \frac{\lambda}{2}\|x\|_2^2, \tag{1}$$

✉ Yifei Wang
wangyf18@stanford.edu

Mert Pilanci
pilanci@stanford.edu

1 Department of Electrical Engineering, Stanford University, 350 Jane Stanford Way, Stanford, CA 94305, USA

where $A \in \mathbb{R}^{n\times d}$ is the data matrix, $b \in \mathbb{R}^{d}$ is the label vector, and $\lambda > 0$ is a regularization parameter.

Typically, one needs to estimate the regularization parameter λ from a set Λ of possible values and select the λ with the best performance on the validation set. For moderate size problem, to obtain an estimate for the regularization parameter λ, one can apply risk estimators such as generalized cross-validation [1], Stein's unbiased risk estimate [2], or unbiased prediction risk estimate [3]. Moreover, solving this problem efficiently on a large scale is of great interest in model selection [4] and transfer learning [5]. For instance, in deep learning-based transfer learning, one needs to tune the last layer of a trained neural network to adapt to a different dataset. Essentially, training the last layer of the neural network is a ridge regression problem using squared loss, where the previous layers of the neural network can be fixed as feature extractors of the raw data.

Classical methods for solving ridge regression include singular value decomposition (SVD) method, warm-started conjugate gradient (CG) method, warm-started preconditioned conjugate gradient (PCG) method, and warm-started iterative Hessian (IHS) sketch method [6]. The SVD method constructs a decomposition of the data matrix and provides a closed-form parameterization of the optimal solution to (1) as a function of λ (see e.g., [7]). The warm-started CG/PCG/IHS method iteratively solves (1) with different values of the regularization parameter λ. They leverage previous solutions along the regularization path as initializers to warm-start the iterations. Besides, for kernel ridge regression, Nyström computational regularization (NCR) [8] applies the Nyström subsampling approaches to reduce the computation cost for calculating the regularization path.

In this paper, we present the iterative Hessian sketch method with binomial decomposition (IHS-BIN) for rapidly solving ridge regression with multiple regularization parameters. The idea is to approximate the linear operator $(A^TA + \lambda I)^{-1}$ via a polynomial of λ constructed by the iterative Hessian sketch method (IHS) [6]. To be specific, IHS is an efficient randomized algorithm for solving large-scale least-square problems. We first focus on the overdetermined case, where $n > d$, and then introduce an extension to the underdetermined case $n \leq d$. Suppose that the size of Λ is T and $\Lambda \subseteq [\lambda_{\min}, \lambda_{\max}]$ with $0 < \lambda_{\min} < \lambda_{\max}$. We compare the computation cost of the proposed IHS-BIN method with other classical solvers for ridge regression in Table 1. For the case where T is large, IHS-BIN with the computation cost of $O(Td)$ is the fastest solver to the best of our knowledge. When T is small, IHS-BIN still offers a substantial improvement in complexity. In Figs. 1 and 2, we present the results on a randomly generated data example and a matrix-valued ridge regression problem with kernel matrix based on the CIFAR-10 dataset. We can observe that IHS-BIN can be significantly faster than other solvers.

This paper is organized as follows. In Sect. 2, we review classical methods for solving ridge regression with multiple λs. As the motivation of IHS-BIN, we introduce gradient descent with binomial decomposition in Sect. 3. We present IHS with binomial decomposition and analyze its convergence rate with different sketching matrices in Sect. 4. In Sect. 5, we also provide a practical method for estimating an appropriate

Table 1 Computational complexity of solving the ridge regression problem (1) for T distinct values of the regularization parameter λ. Here, κ is the condition number of $A^TA + \lambda_{\min}I$, and d_e is effective dimension of $A^TA + \lambda_{\min}I$

Method	small T	large T
IHS-BIN (Ours)	$O\left(d_e^2 d + (dd_e + \mathrm{nnz}(A)) \log\left(\frac{\lambda_{\max}}{\lambda_{\min}}\right)\right)$	$O(Td)$
SVD	$O(nd^2)$	$O(Td^2)$
Warm-started CG	$O(T\mathrm{nnz}(A)\sqrt{\kappa})$	$O(T\mathrm{nnz}(A)\sqrt{\kappa})$
Warm-started PCG	$O(d_e^2 d + \log(d_e)\mathrm{nnz}(A))$	$O(T\mathrm{nnz}(A))$
Warm-started IHS	$O(d_e^2 d + \log(d_e)\mathrm{nnz}(A))$	$O(T\mathrm{nnz}(A))$
NCR	$O(d_e^2 n)$	$O(Td_e^3)$

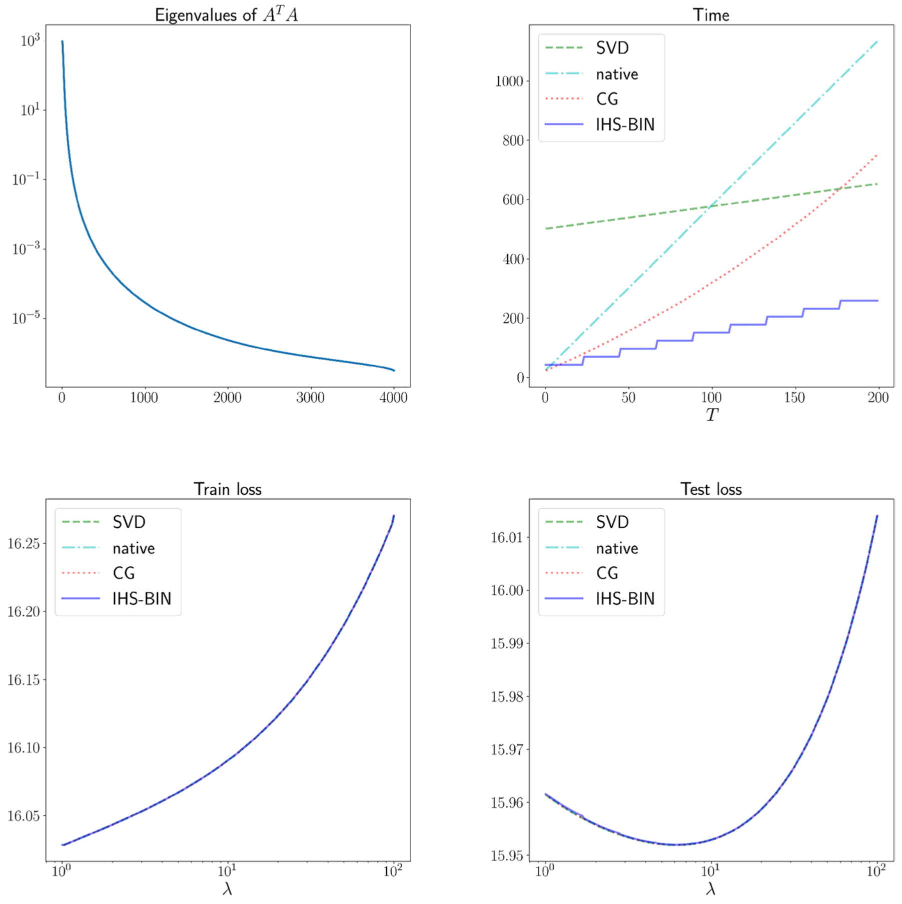

Fig. 1 Training loss, test loss and CPU time on randomly generated data. $n = 20000, d = 4000$. $\lambda_{\min} = 1$. $\lambda_{\max} = 100$. 'native' is the native linear system solver in NumPy. The sketching dimension is set to $m = 1600$

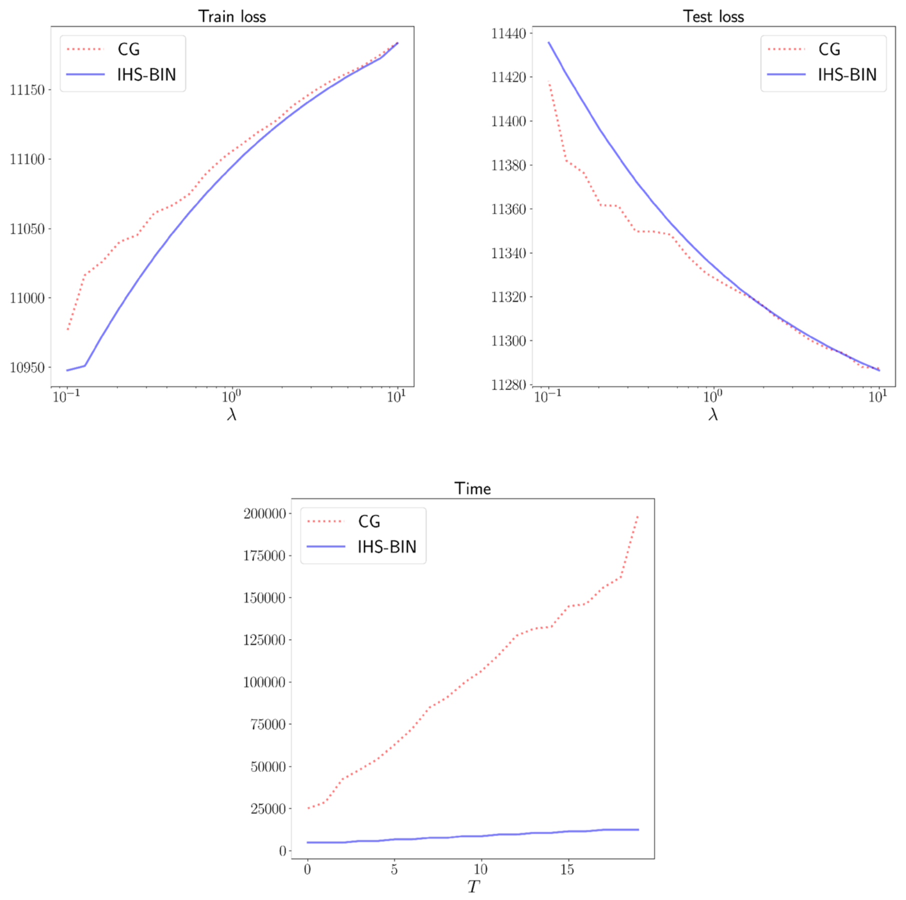

Fig. 2 Training loss, test loss and CPU time on the CIFAR10 dataset. Matrix-valued kernel ridge regression. $n = 25000, d = 25000$. $\lambda_{\min} = 0.1$. $\lambda_{\max} = 10$. We do not calculate the eigenvalues of A^TA since d is large. The sketching dimension is set to $m = 10000$

sketching dimension and extend our algorithm to the under-determined case and matrix-valued ridge regression. The numerical results are presented in Sect. 6.

2 Review of classical methods

We briefly review several classical methods for solving (1) with $\lambda \in \Lambda$, which contains T distinct values. We focus on the case where $n > d$.

2.1 Singular value decomposition

Suppose that the singular value decomposition of A is given by $A = U\Sigma V^T$, where $U \in \mathbb{R}^{n\times d}, \Sigma \in \mathbb{R}^{d\times d}$ and $V \in \mathbb{R}^{d\times d}$. Then, we can calculate the optimal solution to (1) by

$$x^*(\lambda) := (A^TA + \lambda I)^{-1}A^Tb = V(\Sigma^2 + \lambda I)^{-1}\Sigma U^T b$$

The above expression shows that $x^*(\lambda)$ can be computed for all values of λ when the decomposition factors are cached. The total computation cost of the SVD-based computation of the ridge regularization path is therefore $\underbrace{O(nd^2)}_{\text{SVD}} + \underbrace{O(d^2T)}_{\text{updating } \lambda}$.

2.2 Warm-started conjugate gradient method

Suppose that we arrange $\lambda \in \Lambda$ in decreasing order. Then, we can apply the conjugate gradient method to solve (1) with $\lambda \in \Lambda$ from large to small values of λ. We can use the solution for (1) with larger λ as initialization for the next value of λ. From [9], the overall computational cost is given by

$$O(T\text{nnz}(A)\sqrt{\kappa}\log(1/\epsilon)).$$

Here, nnz(A) denotes the number of nonzero elements in A, ϵ is the tolerance of precision to stop the conjugate gradient method, and $\kappa = (\sigma^2_{\max}(A) + \lambda_{\min})/(\sigma^2_{\min}(A) + \lambda_{\min})$ is the largest condition number of $A^TA + \lambda I$ with $\lambda \in \Lambda$. Here, we let $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$ to represent the largest/smallest singular value of A.

2.3 Warm-started preconditioned conjugate gradient method

It is well-known that the number of iterations of warmed-started CG heavily depends on the condition number κ. For ill-conditioned data matrices, the condition number κ may be very large, which leads to slow convergence of the conjugate gradient method, even using the warm-starts. A widely applied approach to deal with the large condition number κ is to apply a randomized preconditioned conjugate gradient (PCG) method [10]. Specifically, we use the random matrix $(A^TS^TSA + \lambda I)^{-1}$ as the preconditioner, where $S \in \mathbb{R}^{m\times n}$ is a sketching matrix. The sketching dimension m is usually proportional to the effective dimension d_e, which will be discussed in detail in Sect. 4.1. To calculate the preconditioner $(A^TS^TSA + \lambda I)^{-1}$ for various λ, we usually compute the SVD of SA, which takes $O(d_e^2 d)$ time. The computational cost of computing SA can be $O(\log(d_e)\text{nnz}(A))$, depending on the type of the sketch as shown in Sect. 4. Given the preconditioner $(A^TS^TSA + \lambda I)^{-1}$, the computational cost of warm-started PCG is given by

$$O(T\text{nnz}(A)\log(1/\epsilon)).$$

2.4 Warm-started iterative Hessian sketch

The iterative Hessian sketch (IHS) method [6] is a randomized sketching method for solving least square problems. The update rule of IHS is as follows:

$$x_{k+1} = x_k - \tau(A^TS^TSA + \lambda I)^{-1}A^T(Ax_k - b + \lambda x_k) \tag{2}$$

Here, $\tau > 0$ is the step size. Similarly, to compute $(A^TS^TSA + \lambda I)^{-1}$ for various λ, usually we compute the SVD of SA, which takes $O(d_e^2 d)$ time. With carefully chosen sketching dimension and sketching matrix, IHS can converge in $O(\log(1/\epsilon))$ iterations. Although IHS is simpler, the computational cost of IHS is similar to the PCG method, which is given by

$$O(T\mathrm{nnz}(A)\log(1/\epsilon)).$$

We note that the above complexity can be high for large data matrices, especially when T, the number of points in the regularization path is also large. In contrast, the proposed approach has complexity $O(Td\log(1/\epsilon))$, which can be significantly smaller when n is large.

3 Gradient descent regularization path

Now, we illustrate the main idea underlying our algorithm. Although this will not be practical, our proposed method is inspired by this approach. Recall that for integer r, the Krylov subspace $\mathcal{K}(r)$ is defined as

$$\mathcal{K}(r) = \mathrm{span}\{A^Tb, (A^TA + \lambda I)A^Tb, \dots, (A^TA + \lambda I)^{r-1}A^Tb\}$$

The gradient descent method with fixed step size for a small number of iterations can be viewed as approximating the minimizer of (1) in the Krylov subspace. Namely, consider the updates

$$\begin{aligned} x_{k+1} &= x_k - \tau\left(A^T(Ax_k - b) + \lambda x_k\right) \\ &= (I - \tau(A^TA + \lambda I))x_k + \tau A^Tb \\ &=: Mx_k + \tau A^Tb. \end{aligned}$$

Here, we denote $M = (I - \tau(A^TA + \lambda I))$. We can express x_k in terms of M and x_0 by

$$x_k = M^kx_0 + \tau M^{k-1}A^Tb + \tau M^{k-2}A^Tb + \cdots + \tau A^Tb.$$

Therefore, we note that $x_{k+1} \in \mathcal{K}(k)$. The binomial expansion formula for M^k is given by

$$M^k = \sum_{j=0}^{k}\binom{k}{j}\lambda^j(-\tau)^j(I - \tau A^TA)^{k-j}.$$

For simplicity, we assume that the iterations are initialized at $x_0 = 0$. Then, we can rewrite x_k as

$$\begin{aligned} x_k &= \tau \sum_{i=0}^{k-1} M^i A^T b \\ &= \tau \sum_{i=0}^{k-1} \sum_{j=0}^{i} \binom{i}{j} \lambda^j (-\tau)^j (I - \tau A^T A)^{i-j} A^T b \\ &= \tau \sum_{j=0}^{k-1} (-\tau\lambda)^j \sum_{i=0}^{k-1-j} \binom{i+j}{j} (I - \tau A^T A)^i A^T b \\ &= \tau \sum_{j=0}^{k-1} (-\tau\lambda)^j u_j. \end{aligned}$$

Here, we denote $u_j = \sum_{i=0}^{k-1-j} \binom{i+j}{j} (I - \tau A^T A)^i A^T b$. Note that the above expression provides an approximate closed form formula for $x(\lambda)$ for all values of λ. More specifically, if we compute $u_0, \ldots, u_{k-1}$, we can instantly compute $x_k = x_k(\lambda)$ for different λ parameters. We call this method GD-BIN and summarize it in Algorithm 1.

```
Input: $A, b, \Lambda = \{\lambda_i\}_{i=1}^T$, iteration number $k$.
Compute $u_j = \sum_{i=0}^{k-1-j} \binom{i+j}{j} (I - \tau A^T A)^i A^T b$ for $j = 0, \ldots, k-1$; for
$i = 1, \ldots, T$ do
    Compose $x_i = \tau \sum_{j=0}^{k-1} (-\tau\lambda_i)^j u_j$;
end
Output: $\{x_i\}_{i=1}^T$
```

Algorithm 1: Gradient descent with binomial decomposition. (GD-BIN)

However, the convergence rate heavily depends on the condition number κ of $A^T A + \lambda_{\min} I$. To obtain an ϵ-approximate solution to (1), it takes approximately $k = O(\log(1/\epsilon)\kappa)$ iterations. The overall computation cost is as follows:

$$\begin{aligned} &\underbrace{O(ndk)}_{\text{compute } (I-\tau A^T A)^i A^T b \text{ and } b_j} + \underbrace{O(Tdk)}_{\text{evaluate } x_k} \\ = \; &\underbrace{O(nd(\log(1/\epsilon)\kappa)}_{\text{compute}(I-\tau A^T A)^i A^T b \text{ and } b_j} + \underbrace{O(Td(\log(1/\epsilon)\kappa)}_{\text{evaluate } x_k} \end{aligned}$$

The main drawback of the gradient descent method is the condition number κ in the computation cost, which is often prohibitively large in practice. It is interesting to note that the dependence on condition number can be improved to $\sqrt{\kappa}$ using conjugate gradient. However, the corresponding regularization path is no longer tractable due to non-constant step-sizes and λ cannot be updated analogously. To circumvent these difficulties and the dependence on κ, we take a different approach and introduce the iterative Hessian sketch (IHS) with binomial decomposition.

4 IHS with binomial decomposition

Suppose that $S \in \mathbb{R}^{m\times n}$ is a sketching matrix, where m is the sketching dimension. IHS [6] employs a randomized Newton direction $((\nabla^2 f)^{1/2})^T S^T S(\nabla^2 f)^{1/2})^{-1}\nabla f(x)$ to minimize the objective function $f(x)$ in (1). The sketching dimension m depends on the effective dimension of $A^TA + \lambda I$, which will be defined later, and it can be significantly smaller than d. Typical choices of sketching matrices include

- Gaussian sketch: Each entry of S follows independent and identically distributed (i.i.d.) Gaussian distribution $\mathcal{N}(0, m^{-1})$.
- CountSketch transform sketch [11]: S is initialized as a matrix of zeros. Then, we set $S_{h(i),i}$ to 1 or -1 with equal probability, where $h(i)$ is chosen from $\{1, \dots, n\}$ uniformly at random.
- Sparse Johnson–Lindenstrauss transform (SJLT) sketch [12]: With column sparsity s, S is constructed by concatenating s independent CountSketch transforms, each of dimension $m/s \times n$.
- Subsampled randomized Hadamard transform (SRHT) sketch [13]: S is a randomized projection matrix.

The update rule of IHS with a fixed regularization parameter is given by

$$\begin{aligned} x_{k+1} &= x_k - \tau(A^TS^TSA + \lambda_0 I)^{-1}A^T(Ax_k - b + \lambda x_k) \\ &= (I - \tau(A^TS^TSA + \lambda_0 I)^{-1}(A^TA + \lambda I)x_k \\ &\quad + \tau(A^TS^TSA + \lambda_0 I)^{-1}A^Tb \\ &= Mx_k + \tau(A^TS^TSA + \lambda_0 I)^{-1}A^Tb. \end{aligned} \tag{3}$$

Here, λ_0 is a fixed parameter. In this case, $M = I - \tau(A^TS^TSA + \lambda_0 I)^{-1}(A^TA + \lambda I)$. Due to the randomized preconditioner, this update can be viewed as a better approximation of the optimal solution than the one in the Krylov subspace $\mathcal{K}(k+1)$ in a similar spirit to rational Krylov subspace methods [14]. Rational Krylov methods apply rational functions to the matrix vector products to construct rational Krylov subspaces. The use of randomized preconditioners and the binomial decomposition over the regularization path is a novel idea to the best of our knowledge.

Based on the update rule of IHS, we can express x_k in terms of M and x_0 via

$$\begin{aligned} x_k = M^k x_0 &+ \tau M^{k-1}(A^TS^TSA + \lambda_0 I)^{-1}A^Tb \\ &+ \cdots + \tau(A^TS^TSA + \lambda_0 I)^{-1}A^Tb. \end{aligned}$$

For simplicity, we also assume that the iterations are initialized at $x_0 = 0$. We denote $P_S = (A^TS^TSA + \lambda_0 I)^{-1}$.

Proposition 1 *We can express x_k as a polynomial function of λ as follows*:

$$x_k = \tau \sum_{j=0}^{k-1} (\tau\lambda)^j \tilde{u}_j,$$

Here, $\tilde{u}_j = \sum_{i=j}^{k-1} u_{i,j}$, *where* $u_{i,j} \in \mathbb{R}^d$ *can be recursively computed via*

$$\begin{aligned} u_{i+1,0} &= (I - \tau P_S A^T A) u_{i,0}, \quad u_{i+1,i+1} = -P_S u_{i,i}, \\ u_{i+1,j} &= (I - \tau P_S A^T A) u_{i,j} - P_S u_{i,j-1}, \quad 1 \le j \le i, \end{aligned} \tag{4}$$

with the initial condition $u_{0,0} = P_S A^T b$.

Proof From the previous derivation, by taking $x_0 = 0$, we note that

$$x_k = \sum_{i=0}^{k-1} \tau M^i P_S A^T b.$$

Therefore, it is sufficient to show that $M^i P_S A^T b$ can be written as a polynomial function of λ. We first claim that for all integer $i \ge 0$,

$$M^i P_S A^T b = \sum_{j=0}^{i} (\tau\lambda)^j u_{i,j}. \tag{5}$$

We prove this claim by mathematical induction. It is easy to observe that (5) holds for $i = 0$. Suppose that (5) holds for i. For $i + 1$, we note that

$$\begin{aligned} M^{i+1} P_S A^T b &= M(M^i P_S A^T b) \\ &= \sum_{j=0}^{i} \big((\tau\lambda)^j (I - P_S A^T A) u_{i,j} - (\tau\lambda)^{j+1} P_S u_{i,j} \big) \\ &= \sum_{j=1}^{i} (\tau\lambda)^j \big((I - P_S A^T A) u_{i,j} - P_S u_{i,j-1} \big) \\ &\quad + (I - P_S A^T A) u_{i,0} - (\tau\lambda)^{i+1} P_S u_{i,i} \\ &= \sum_{j=0}^{i+1} (\tau\lambda)^j u_{i+1,j}. \end{aligned}$$

Hence, (5) also holds for $i + 1$.

As a result, we can easily compute that

$$x_k = \sum_{i=0}^{k-1} \tau M^i P_S A^T b = \tau \sum_{i=0}^{k-1} \sum_{j=0}^{i} (\tau\lambda)^j u_{i,j} = \tau \sum_{j=0}^{k-1} (\tau\lambda)^j \tilde{u}_j.$$

□

To compute $(A^TS^TSA + \lambda_0 I)^{-1}$, we perform the singular value decomposition on SA, i.e., $SA = U_1\Sigma_1 V_1$. Suppose that $m < d$. Then, we have

$$(A^TS^TSA + \lambda_0 I)^{-1} = V_1^T(\Sigma_1^2 + \lambda_0 I)^{-1}V_1 + \lambda_0^{-1}(I - V_1^T V_1).$$

Thus, for an arbitrary $v \in \mathbb{R}^d$, we have

$$(A^TS^TSA + \lambda_0 I)^{-1}v = v/\lambda_0 + V_1^T((\Sigma_1^2 + \lambda_0 I)^{-1} - \lambda_0^{-1})V_1 v.$$

For $m \geq d$, then, we have

$$(A^TS^TSA + \lambda_0 I)^{-1}v = V_1^T(\Sigma_1^2 + \lambda_0 I)^{-1}V_1 v.$$

```
Input: $A, b, P_S, \tau, k$.
Set $u_i = 0$ and $\tilde{u}_i = 0$ with $i = 0, \dots, k-1$;
Calculate $u_0 = -P_S A^T b$;
Let $\tilde{u}_0 = \tilde{u}_0 + u_0$;
for $i = 1$ to $k - 1$ do
    Calculate $u_i = -P_S u_{i-1}$;
    for $j = i - 1$ to $1$ do
        Calculate $u_j = u_j - P_S(\tau A^T A u_j + u_{j-1})$;
    end
    Calculate $u_0 = u_0 - P_S \tau A^T A u_0$;
    Update $\tilde{u}_j = \tilde{u}_j + u_j$ for $j = 0, \dots, i$;
end
Output: $\{\tilde{u}_j\}_{j=0}^{k-1}$
```

Algorithm 2: Calculation of basis $\tilde{u}_0, \dots, \tilde{u}_{k-1}$ for binomial decomposition.

We summarize the calculation of $\tilde{u}_0, \dots, \tilde{u}_{k-1}$ in Algorithm 2. For computing the sketching SA, we list the computation cost for different sketching matrix as follows:

- Gaussian sketch: $O(mnd)$ or $O(m\text{nnz}(A))$ for a sparse matrix.
- SRHT sketch: $O(\log(m)nd)$ or $O(\log(m)\text{nnz}(A))$ for a sparse matrix.
- SJLT sketch: $O(snd)$ or $O(s\text{nnz}(A))$ for a sparse matrix. Here, s is the sparsity of SJLT sketch.

As the sketching dimension is proportional to the effective dimension d_e, which can be significantly smaller than d, here we assume that $m < d$. Hence, the computation cost to compute the SVD of SA is $O(m^2d)$. Ignoring the complexity of sketching, the computation cost of IHS-BIN is as follows:

$$\underbrace{O(m^2d)}_{\text{SVD of } SA} + \underbrace{O((md + nd)k^2)}_{\text{compute } \tilde{u}_j} + \underbrace{O(Tdk)}_{\text{evaluate } x_k}.$$

For a sparse matrix A, the computation cost reduces to

$$\underbrace{O(m^2 d)}_{\text{SVD of } SA} + \underbrace{O((md + \mathrm{nnz}(A))k^2)}_{\text{compute } \tilde{u}_j} + \underbrace{O(Tdk)}_{\text{evaluate } x_k}.$$

The overall algorithm is summarized in Algorithm 3.

```
Input: $A, b, \Lambda = \{\lambda_i\}_{i=1}^T$, iteration number $k$.
Generate the sketching matrix $S$ and compute the SVD of $SA$; for
 $i = 1, \dots, T$ do
    Compute $\{\tilde{u}_j\}_{j=0}^{k-1}$ using Algorithm 2; Compose
     $x_i = \tau \sum_{j=0}^{k-1} (\tau\lambda_i)^j \tilde{u}_j$;
end
Output: $\{x_i\}_{i=1}^T$
```

Algorithm 3: Iterative Hessian Sketch with binomial decomposition. (IHS-BIN)

4.1 Convergence analysis

In this subsection, we analyze the convergence rate of IHS with binomial decomposition. We first introduce some notations. Suppose that $A = U\Sigma V^T$ is the singular value decomposition, and let $\sigma_1 \geq \cdots \geq \sigma_d$ denote singular values of A. Let $\bar{A} = \begin{bmatrix} A \\ \sqrt{\lambda_0} I, \end{bmatrix}$ $\tilde{A} = \begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix}$. Let $\bar{U}$ be a matrix of left singular vectors of $\bar{A}$. For $\lambda_0 \geq 0$, we introduce a diagonal matrix $D = \mathrm{diag}\left(\frac{\sigma_1}{\sqrt{\sigma_1^2+\lambda_0}}, \dots, \frac{\sigma_d}{\sqrt{\sigma_d^2+\lambda_0}} \right)$ and define the effective dimension by

$$d_e = \frac{\|D\|_F^2}{\|D\|_2^2}. \tag{6}$$

This will influence the sketching dimension, which will be discussed in detail later. Define

$$\bar{\Sigma} = \mathrm{diag}\left(\sqrt{\sigma_1^2 + \lambda_0}, \dots, \sqrt{\sigma_d^2 + \lambda_0} \right),$$
$$\tilde{\Sigma} = \mathrm{diag}\left(\sqrt{\sigma_1^2 + \lambda}, \dots, \sqrt{\sigma_d^2 + \lambda} \right).$$

Denote $\bar{S} = \begin{bmatrix} S & 0 \\ 0 & I_{\mathrm{d}} \end{bmatrix}$. We define two matrices

$$\tilde{C}_S = \tilde{\Sigma}^{-1} \bar{\Sigma} C_S \bar{\Sigma} \tilde{\Sigma}^{-1}, \quad C_S = \bar{U}^T \bar{S}^T \bar{S} \bar{U}.$$

Denote γ_1, γ_d to be the largest/smallest eigenvalue of C_S. For two real numbers $\rho_1 > \rho_2 > 0$, we define the S-measurable event $\mathcal{E}_S = \{\rho_2 \leq \gamma_d \leq \gamma_1 \leq \rho_1\}$. We evaluate the error by $\delta_k = \frac{1}{2}\|\tilde{A}(x_k - x^*)\|^2$.

Theorem 2 *Suppose that we solve (1) for $\lambda \in [\lambda_{\min}, \lambda_{\max}]$. Denote $\tilde{\kappa} = \frac{\rho_1 \lambda_{\max}}{\rho_2 \lambda_{\min}}$. By setting $\lambda_0 = \sqrt{\lambda_{\max}\lambda_{\min}}$ and taking $\alpha = 2\sqrt{\rho_1^{-1}\rho_2^{-1}}/(\sqrt{\tilde{\kappa}} + \sqrt{\tilde{\kappa}^{-1}})$, then, conditioned on the event $\mathcal{E}_S$, the IHS-BIN satisfies that at each iteration,*

$$\frac{\delta_{k+1}}{\delta_k} \leq \left(\frac{\tilde{\kappa} - 1}{\tilde{\kappa} + 1}\right)^2.$$

Remark 1 If we can estimate the smallest singular value σ_d of A, we can refine the convergence rate as follows. Denote $\hat{\kappa} = \frac{\rho_1(\lambda_{\max} + \sigma_d^2)}{\rho_2(\lambda_{\min} + \sigma_d^2)}$. Assume that we set $\lambda_0 = \sqrt{(\lambda_{\max} + \sigma_d^2)(\lambda_{\min} + \sigma_d^2)} - \sigma_d^2$ and take a constant step size $\alpha = 2\sqrt{\rho_1^{-1}\rho_2^{-1}}/(\sqrt{\hat{\kappa}} + \sqrt{\hat{\kappa}^{-1}})$. Then, conditioned on the event $\mathcal{E}_S$, the IHS-BIN satisfies that at each iteration

$$\frac{\delta_{k+1}}{\delta_k} \leq \left(\frac{\hat{\kappa} - 1}{\hat{\kappa} + 1}\right)^2.$$

We also note that the convergence only depends on the event $\mathcal{E}_S$. In other words, as long as the event $\mathcal{E}_S$ is satisfied, we have the convergence guarantee for solving every ridge regression problem along the entire regularization path.

Denote $\beta = \lambda_{\max}\lambda_{\min}^{-1}$. Thus, to reach an ϵ precision solution, it takes $k = O(\log(1/\epsilon)\tilde{\kappa}) = O(\log(1/\epsilon)\beta)$ iterations. When β is large, the iteration number k can be large, which may not be efficient.

To deal with this problem, we split $[\lambda_{\min}, \lambda_{\max}]$ into L smaller intervals

$$[\lambda^{(0)}, \lambda^{(1)}], [\lambda^{(1)}, \lambda^{(2)}], \dots, [\lambda^{(L-1)}, \lambda^{(L)}].$$

Here, we let $\lambda^{(i)} = \lambda_{\min}\beta^{i/L}$. For each small interval $[\lambda^{(i)}, \lambda^{(i+1)}]$, it takes approximately $k = \mathcal{O}(\log(1/\epsilon)\beta^{1/L})$ iterations to reach an ϵ precision solution. Compared to the computation cost in computing basis $\tilde{u}_j$, the computation cost of computing SA and performing SVD of SA is negligible. Hence, the major computation cost is as follows:

$$\underbrace{O(L(md + nd)\log(1/\epsilon)^2\beta^{2/L})}_{\text{compute } \tilde{u}_j \ L \text{ times}} + \underbrace{O(Td\log(1/\epsilon)\beta^{1/L})}_{\text{evaluate } x_k}.$$

Suppose that we take $L = \lfloor 2\log\beta \rfloor$. Then, the computation cost writes

$$\underbrace{O((md + nd)\log\beta\log(1/\epsilon)^2)}_{\text{compute } \tilde{u}_j \ L \text{ times}} + \underbrace{O(Td\log(1/\epsilon))}_{\text{evaluate } x_k}.$$

Similarly, if A is a sparse matrix, then the computation cost is as follows:

$$\underbrace{O((md+\mathrm{nnz}(A))\log\beta\log(1/\epsilon)^2)}_{\text{compute } \tilde{u}_j\ L \text{ times}} + \underbrace{O(Td\log(1/\epsilon))}_{\text{evaluate } x_k}.$$

Namely, we can improve the dependence of β in computing binomial basis $\tilde{u}_j$ from β^2 to $\log\beta$.

4.2 Proof of theorem 2

For the update rule, we have

$$x_{k+1} = x_k - \alpha\left(\bar{A}^T\bar{S}^T\bar{S}\bar{A}\right)^{-1}\tilde{A}^T(\tilde{A}x_k - \bar{b}).$$

We use the notation $e_t = \tilde{U}^T\tilde{A}(x_k - x^*)$. Here, x^* is the unique minimizer of

$$\min_{x\in\mathbb{R}^d} \frac{1}{2}\|Ax-b\|_2^2 + \frac{\lambda}{2}\|x\|_2^2. \tag{7}$$

We obtain that

$$\begin{aligned} e_{k+1} &= e_k - \alpha\tilde{U}^T\tilde{A}\left(\bar{A}^T\bar{S}^T\bar{S}\bar{A}\right)^{-1}\tilde{A}^T\tilde{U}e_k \\ &= \left(I - \alpha\tilde{\Sigma}(\bar{\Sigma}\bar{U}^T\bar{S}^T\bar{S}\bar{U}\bar{\Sigma})^{-1}\tilde{\Sigma}\right)e_k \\ &= (I - \alpha\tilde{C}_S^{-1})e_k. \end{aligned}$$

We note that

$$\begin{aligned} \sqrt{\delta_{k+1}} &= \|e_{k+1}\|_2 \le \|I - \alpha\tilde{C}_S^{-1}\|_2\|e_k\|_2 \\ &= \|I - \alpha\tilde{C}_S^{-1}\|_2\sqrt{\delta_k}. \end{aligned}$$

Then, the convergence rate depends on the condition number of $\tilde{C}_S$. For the rest of the proof, we assume that the event $\mathcal{E}_S$ holds. Based on the estimations ρ_1, ρ_2 for the extreme eigenvalues of C_S, we define $\tilde{\rho}_1(\lambda) > \tilde{\rho}_2(\lambda) > 0$ as follows: if $\lambda \ge \lambda_0$, we let

$$\tilde{\rho}_2(\lambda) = \frac{\sigma_d^2+\lambda_0}{\sigma_d^2+\lambda}\rho_2, \quad \tilde{\rho}_1(\lambda) = \frac{\sigma_1^2+\lambda_0}{\sigma_1^2+\lambda}\rho_1.$$

Otherwise, if $\lambda \le \lambda_0$, we let

$$\tilde{\rho}_2(\lambda) = \frac{\sigma_1^2+\lambda_0}{\sigma_1^2+\lambda}\rho_2, \quad \tilde{\rho}_1(\lambda) = \frac{\sigma_d^2+\lambda_0}{\sigma_d^2+\lambda}\rho_1.$$

Hence, the extreme eigenvalues of $\tilde{C}_S$ is bounded in $[\tilde{\rho}_2(\lambda), \tilde{\rho}_1(\lambda)]$. Hence, the convergence rate is as follows:

$$\sqrt{\frac{\delta_{k+1}}{\delta_k}} \le \max\{|1-\tilde{\alpha}\rho_1(\lambda)^{-1}|, |1-\alpha\tilde{\rho}_2(\lambda)^{-1}|\}.$$

For $\lambda \in [\lambda_{\min}, \lambda_{\max}]$, we want to minimize the worst convergence rate:

$$\min_{\alpha>0,\lambda_0\in[\lambda_{\min},\lambda_{\max}]} \max_{\lambda\in[\lambda_{\min},\lambda_{\max}]} \max\{|1-\alpha\tilde{\rho}_1(\lambda)^{-1}|, |1-\alpha\tilde{\rho}_2(\lambda)^{-1}|\}.$$

For $\lambda \geq \lambda_0$, we have

$$\tilde{\rho}_2(\lambda) = \frac{\sigma_d^2+\lambda_0}{\sigma_d^2+\lambda}\rho_2, \quad \tilde{\rho}_1(\lambda) = \frac{\sigma_1^2+\lambda_0}{\sigma_1^2+\lambda}\rho_1,$$

which yields

$$\tilde{\rho}_2(\lambda_{\max}) \leq \tilde{\rho}_2(\lambda) \leq \tilde{\rho}_1(\lambda) \leq \tilde{\rho}_1(\lambda_{\max}).$$

This indicates that

$$\begin{aligned}&\max_{\lambda\in[\lambda_0,\lambda_{\max}]} \max\{|1-\alpha\tilde{\rho}_1(\lambda)^{-1}|, |1-\alpha\tilde{\rho}_2(\lambda)^{-1}|\}\\ &= \max\{|1-\alpha\tilde{\rho}_1(\lambda_{\max})^{-1}|, |1-\alpha\tilde{\rho}_2(\lambda_{\max})^{-1}|\}.\end{aligned}$$

For $\lambda \leq \lambda_0$, similarly, we have

$$\tilde{\rho}_2(\lambda) = \frac{\sigma_1^2+\lambda_0}{\sigma_1^2+\lambda}\rho_2, \quad \tilde{\rho}_1(\lambda) = \frac{\sigma_d^2+\lambda_0}{\sigma_d^2+\lambda}\rho_1.$$

This indicates that

$$\begin{aligned}&\max_{\lambda\in[\lambda_{\min},\lambda_0]} \max\{|1-\alpha\tilde{\rho}_1(\lambda)^{-1}|, |1-\alpha\tilde{\rho}_2(\lambda)^{-1}|\}\\ &= \max\{|1-\alpha\tilde{\rho}_1(\lambda_{\min})^{-1}|, |1-\alpha\tilde{\rho}_2(\lambda_{\min})^{-1}|\}.\end{aligned}$$

We further notice that

$$\tilde{\rho}_2(\lambda_{\max}) \leq \rho_2 \leq \tilde{\rho}_2(\lambda_{\min}), \quad \tilde{\rho}_1(\lambda_{\max}) \leq \rho_1 \leq \tilde{\rho}_1(\lambda_{\min}).$$

In short, we have

$$\begin{aligned}&\max_{\lambda\in[\lambda_{\min},\lambda_{\max}]} \max\{|1-\alpha\tilde{\rho}_1(\lambda)^{-1}|, |1-\alpha\tilde{\rho}_2(\lambda)^{-1}|\}\\ &= \max\left\{\left|1-\tilde{\rho}_2(\lambda_{\max})^{-1}\alpha\right|, \left|1-\tilde{\rho}_1(\lambda_{\min})^{-1}\alpha\right|\right\}\\ &= \max\left\{\left|1-\frac{\sigma_d^2+\lambda_{\max}}{\sigma_d^2+\lambda_0}\rho_2^{-1}\alpha\right|, \left|1-\frac{\sigma_d^2+\lambda_{\min}}{\sigma_d^2+\lambda_0}\rho_1^{-1}\alpha\right|\right\}.\end{aligned}$$

The above quantity is minimized when

$$\alpha/(\lambda_0+\sigma_d^2) = 2\big((\sigma_d^2+\lambda_{\min})\rho_1^{-1} + (\sigma_d^2+\lambda_{\max})\rho_2^{-1}\big)^{-1}.$$

Thus, by taking $\lambda_0 = \sqrt{(\lambda_{\max} + \sigma_d^2)(\lambda_{\min} + \sigma_d^2)} - \sigma_d^2$, the optimal step size follows $\alpha = 2\sqrt{\rho_1^{-1}\rho_2^{-1}}/\left(\sqrt{\kappa} + \sqrt{\kappa^{-1}}\right)$. In summary, we have

$$
\begin{aligned}
\sqrt{\frac{\delta_{k+1}}{\delta_k}} \le & \min_{\alpha>0,\lambda_0\in[\lambda_{\min},\lambda_{\max}]} \max_{\lambda\in[\lambda_{\min},\lambda_{\max}]} \\
& \max\{|1-\alpha\tilde{\rho}_1(\lambda)^{-1}|, |1-\alpha\tilde{\rho}_2(\lambda)^{-1}|\} \\
= & \frac{\kappa-1}{\kappa+1}.
\end{aligned}
$$

For the case where σ_d is unknown, since

$$
\frac{\lambda_{\max}}{\lambda_0}\rho_2^{-1} \ge \frac{\sigma_d^2+\lambda_{\max}}{\sigma_d^2+\lambda_0}\rho_2^{-1} \ge \frac{\sigma_d^2+\lambda_{\min}}{\sigma_d^2+\lambda_0}\rho_1^{-1} \ge \frac{\lambda_{\min}}{\lambda_0}\rho_1^{-1},
$$

we can relax the bound as follows:

$$
\begin{aligned}
& \max\left\{\left|1-\frac{\sigma_d^2+\lambda_{\max}}{\sigma_d^2+\lambda_0}\rho_2^{-1}\alpha\right|, \left|1-\frac{\sigma_d^2+\lambda_{\min}}{\sigma_d^2+\lambda_0}\rho_1^{-1}\alpha\right|\right\} \\
& \le \max\left\{\left|1-\frac{\lambda_{\max}}{\lambda_0}\rho_2^{-1}\alpha\right|, \left|1-\frac{\lambda_{\min}}{\lambda_0}\rho_1^{-1}\alpha\right|\right\}.
\end{aligned}
$$

Similarly, the above quantity isminimized when $\alpha/(\lambda_0) = 2\left(\lambda_{\min}\rho_1^{-1} + \lambda_{\max}\rho_2^{-1}\right)^{-1}$. For $\lambda_0 = \sqrt{\lambda_{\min}\lambda_{\max}}$, the optimal step size writes $\alpha = 2\sqrt{\rho_1^{-1}\rho_2^{-1}}/\left(\sqrt{\tilde{\kappa}} + \sqrt{\tilde{\kappa}^{-1}}\right)$. We then have

$$
\begin{aligned}
\sqrt{\frac{\delta_{k+1}}{\delta_k}} \le & \min_{\alpha>0,\lambda_0\in[\lambda_{\min},\lambda_{\max}]} \max_{\lambda\in[\lambda_{\min},\lambda_{\max}]} \\
& \max\{|1-\alpha\tilde{\rho}_1(\lambda)^{-1}|, |1-\alpha\tilde{\rho}_2(\lambda)^{-1}|\} \\
= & \frac{\tilde{\kappa}-1}{\tilde{\kappa}+1}.
\end{aligned}
$$

4.3 Sharp estimates of extreme eigenvalues of C_S

We review several sharp estimates of γ_1, γ_d and discuss the probability that $\mathcal{E}_S$ holds. For the Gaussian case, we have the following theorem introduced in [15].

Theorem 3 *Suppose that $S \in \mathbb{R}^{m\times n}$ is a Gaussian sketching matrix. Consider d_e defined in (6) and a parameter $\rho \in (0,1)$. If $m \ge d_e/\rho$, then for any $\eta \in (0,(1-\sqrt{\rho})^2/4)$, with $c(\eta) = \left(\frac{1+\sqrt{\eta}}{1-\sqrt{\eta}}\right)^2$ and*

$$\begin{cases} \rho_1 = 1 - \|D\|_2^2 + \|D\|_2^2(1+\sqrt{\rho})^2(1+\sqrt{\eta})^2, \\ \rho_2 = 1 - \|D\|_2^2 + \|D\|_2^2\left(1-\sqrt{c(\eta)\rho}\right)^2, \end{cases}$$

the event $\mathcal{E}_S$ *holds with probability at least* $1 - 16e^{-\eta^2\rho m/2}$

For the SJLT sketching, following Lemma 3.3 in [16], we have an estimate on ρ_1 and ρ_2.

Theorem 4 *Suppose that* $S \in \mathbb{R}^{m\times n}$ *is an SJLT sketching matrix with sparsity*

$$s = \Omega(\log_\alpha(d_e/\delta)/\epsilon) \tag{8}$$

in each column where $\alpha > 2, \delta < 1/2, \epsilon < 1/2$. *If the sketch size satisfies*

$$m = \Omega(\alpha d_e \log_\alpha(d_e/\delta)/\epsilon^2),$$

then, with probability at least $1-\delta$, *the event* $\mathcal{E}_S$ *holds.*

For the SRHT case, we introduce the relevant factor

$$C(m, d_e) = \frac{16}{3}\left(1 + \sqrt{\frac{8\log(d_e n)}{d_e}}\right)^2.$$

The following theorem in [15] provides a sharp estimate on ρ_1 and ρ_2.

Theorem 5 *Suppose that* $S \in \mathbb{R}^{m\times n}$ *is an SRHT sketching matrix. Consider* d_e *defined in* (6) *and a parameter* $\rho \in (0,1)$. *If* $m \geq C(n, d_e)\frac{d_e\log(d_e)}{\rho}$. *Then, it holds with probability at least* $9/d_e$ *such that* $\mathcal{E}_S$ *holds with* $\rho_1 = 1 + \|D\|_2^2\rho$ *and* $\rho_2 = 1 - \|D\|_2^2\rho$.

5 Estimation of the effective dimension and extensions of IHS-BIN

5.1 Estimation of the effective dimension

From previous theorems, an appropriate sketching size depends on the effective dimension d_e. Nevertheless, in practice, usually the estimation of d_e is available when d_e is small, see [17]. Following the adaptive method described in [15], we propose a practical method for finding an appropriate sketching size.

We apply IHS to solve (1) with $\lambda = \lambda_{\min}$. In k-th iteration, we first calculate the direction

$$d_k = (A^T S^T S A + \lambda_{\min} I)^{-1} A^T (A x_k - b + \lambda_{\min} x_k).$$

Then, we calculate a step size $\tau_k = \gamma_1^j$ satisfying the Armijo condition. Namely, j is the smallest non-negative integer satisfying

$$f(x_k - \gamma_1^j d_k) \le f(x_k) - \gamma_2 \gamma_1^j d_k^T \nabla f(x_k). \tag{9}$$

Here, we write $f(x) = \frac{1}{2}\|Ax - b\|_2^2 + \frac{\lambda_{\min}}{2}\|x\|_2^2$ and $\gamma_1, \gamma_2 \in (0,1)$ are parameters. Then, we update $x_{k+1} = x_k - \tau_k d_k$.

We evaluate the following quantity per iteration:

$$\tilde{\delta}_k = d_k^T A^T (A x_k - b + \lambda_{\min} x_k).$$

If $\tilde{\delta}_{k+1} \ge \gamma_3 \delta_k$ for some $\gamma_3 > 0$, then we let $m = 2m$ and sample the sketching matrix. We stop the algorithm when $\tilde{\delta}_k < \epsilon$ for some $\epsilon > 0$. The whole algorithm is described in Algorithm 4. In numerical experiment, we set $\gamma_3 = 0.5$.

```
Input: $A, b, x_0, \lambda_{\min}, \epsilon, \gamma_1, \gamma_2, \gamma_3$.
Set $k = 0$. Compute $d_0$ and $\tilde{\delta}_0$;
while $\tilde{\delta}_k > \epsilon$ do
    Calculate a step size $\tau_k = \gamma_1^j$ satisfying the Armijo condition (9);
    Update $x_{k+1} = x_k - \tau_k d_k$;
    Calculate $d_{k+1}$ and $\tilde{\delta}_{k+1}$;
end
if $\tilde{\delta}_{k+1} \ge \gamma_3 \delta_k$ then
    Set $m = 2m$ and sample the sketching matrix $S$;
    Recompute $d_k$ based on $S$;
else
    Set $k = k + 1$;
end
Output: $m$
```

Algorithm 4: Adaptive estimation of sketching dimension

5.2 Extension to the under-determined case

For the under-determined case where $n < d$, we consider the dual problem of (1):

$$\min_{z \in \mathbb{R}^m} \frac{1}{2}\|A^T z\|^2 + \frac{\lambda}{2}\|z\|^2 - b^T z. \tag{10}$$

The optimal solution z to the dual problem is related to the optimal solution to the primal problem by

$$v = A^T z.$$

Hence, we can apply similar methods in the under-parameterized case to solve (10).

5.3 Extension for matrix-valued ridge-regression

Consider the following matrix-valued ridge-regression problem:

$$\min_{x\in\mathbb{R}^{d\times K}} \frac{1}{2}\|AX-B\|_F^2+\frac{\lambda}{2}\|X\|_F^2, \tag{11}$$

where $A\in\mathbb{R}^{n\times d}$ and $B\in\mathbb{R}^{n\times K}$. We can easily extend IHS-BIN for solving this problem.

```
Input: $A, B, P_S, \tau, k$.
Set $U_i = 0$ and $\tilde{U}_i = 0$ with $i = 0, \dots, k-1$;
Calculate $U_0 = -P_S A^T B$;
Let $\tilde{U}_0 = \tilde{U}_0 + U_0$;
for $i = 1$ to $k-1$ do
    Calculate $U_i = -P_S U_{i-1}$;
    for $j = i-1$ to $1$ do
        Calculate $U_j = U_j - P_S(\tau A^T A U_j + U_{j-1})$;
    end
    Calculate $U_0 = U_0 - P_S \tau A^T A U_0$;
    Update $\tilde{U}_j = \tilde{U}_j + U_j$ for $j = 0, \dots, i$;
end
Output: $\{\tilde{U}_j\}_{j=0}^{k-1}$
```

Algorithm 5: Calculation of basis $\tilde{u}_0, \dots, \tilde{u}_{k-1}$ for IHS-BIN with matrix-valued ridge regression.

```
Input: $A, b, \Lambda = \{\lambda_i\}_{i=1}^T$, iteration number $k$.
Generate the sketching matrix $S$ and compute the SVD of $SA$;
for $i = 1, \dots, T$ do
    Compute $\{\tilde{U}_j\}_{j=0}^{k-1}$ using Algorithm 5;
    Compose $X_i = \tau \sum_{j=0}^{k-1} (\tau\lambda_i)^j \tilde{U}_j$;
end
Output: $\{X_i\}_{i=1}^T$
```

Algorithm 6: IHS-BIN with matrix-valued ridge regression.

Neglecting the computation cost of computing SA, the computation cost of IHS-BIN becomes

$$\underbrace{O(m^2d)}_{\text{SVD of } SA}+\underbrace{O(K(md+nd)\log\beta\log(1/\epsilon)^2)}_{\text{compute } \tilde{u}_j\ L \text{ times}}+\underbrace{O(TKd\log(1/\epsilon))}_{\text{evaluate } x_k},$$

for a dense matrix A, or

$$\underbrace{O(m^2d)}_{\text{SVD of } SA}+\underbrace{O(K(md+\text{nnz}(A))\log\beta\log(1/\epsilon)^2)}_{\text{compute } \tilde{u}_j\ L \text{ times}}+\underbrace{O(TKd\log(1/\epsilon))}_{\text{evaluate } x_k},$$

for a sparse matrix A. For comparison, the computation cost of SVD-based method is as follows:

$$\underbrace{O(n^2d)}_{\text{SVD}}+O(d^2KT).$$

The computation cost of CG-based method writes

$$O(TKnd\sqrt{\kappa}\log(1/\epsilon)) \text{ or } O(TK\text{nnz}(A)\sqrt{\kappa}\log(1/\epsilon)).$$

6 Numerical results

In this section, we present numerical comparisons between IHS-BIN and other methods for solving least-square problems with various regularization parameters. We test over randomly generated data and real data. For real data, we collect datasets from LIBSVM[1] under the modified BSD license. We randomly split half of the data matrix as the training data A and the other as the test data matrix $\tilde{A}$. We denote b and $\tilde{b}$ as the corresponding labels of A and $\tilde{A}$. All numerical experiments are conducted on a Dell PowerEdge R840 workstation (64 core, 3TB ram). We provide numerical comparisons with the following baseline algorithms. **SVD:** the singular value decomposition-based method., **native:** the native linear system solver in NumPy, **CG:** warm-started conjugate gradient method. For IHS-BIN, we use the SJLT sketching matrices with sparsity 1. In IHS-BIN, we split the interval $[\lambda_{\min},\lambda_{\max}]$ into $L=\lfloor 2\log(\lambda_{\max}/\lambda_{\min})\rfloor$ small intervals. For each interval $[\lambda^{(i)},\lambda^{(i+1)}]$, we update the set size τ by using the backtracking line search with respect to the problem with regularization parameter $\lambda=\lambda^{(i)}$. The source code is available publicly and can be found in https://github.com/pilancilab/IHS-BIN.

6.1 Datasets setup in the numerical experiments

For the randomly generated data, each row of A follows $\mathcal{N}(0,\Sigma^2)$, where $\Sigma_{i,j}=\frac{1}{nd}\alpha^{|i-j|}$. Here, we let $\alpha=0.99$. The training loss and the test loss are as follows:

$$L_{\text{train}}=\frac{1}{2}\|Av-b\|_2^2+\frac{\lambda}{2}\|v\|^2,\quad L_{\text{test}}=\frac{1}{2}\|\tilde{A}v-\tilde{b}\|_2^2.$$

Here, each row of $\tilde{A}$ follows $\mathcal{N}(0,\Sigma^2)$. And we let

[1] https://www.csie.ntu.edu.tw/~cjlin/libsvm/.

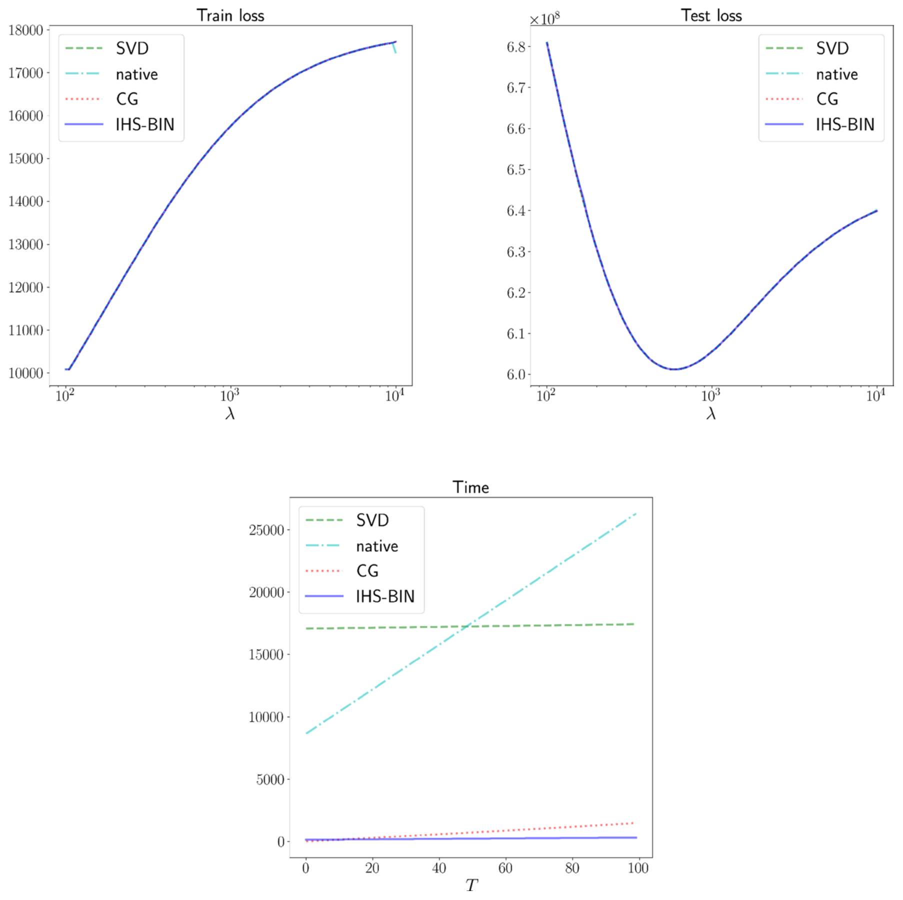

Fig. 3 Training loss, test loss and time. Real vs. Simulated (real-sim). $n = 36000, d = 20958, m = 8000$. $\lambda_{\min} = 100$. $\lambda_{\max} = 10^4$. We do not calculate the eigenvalues of $A^T A$ since d is large

$$b = Av^* + \eta, \quad \tilde{b} = \tilde{A}v^* + \tilde{\eta}.$$

Here, $v^* \sim \mathcal{N}(0, I_d/d)$ and $\eta, \tilde{\eta} \sim \mathcal{N}(0, \sigma^2)$ where $\sigma > 0$ is a parameter.

For the randomly generated data with clustered eigenvalue, each row of A follows $\mathcal{N}(0, \Sigma^2)$, where Σ is a diagonal matrix whose first $d/2$ entries follow $\sigma_i = 2^{u_i}$ and the last $d/2$ entries follow $\sigma_i = 0.01 * 2^{u_i}$. Here, u_i is uniformly randomly drawn from $[-1, 1]$. The setup of label matrix b and test data pair $\tilde{A}, \tilde{b}$ is similar as the previous case.

For real data, we linearly rescale the entry of A into $[-1, 1]$. Besides, for CIFAR10, we use the kernel matrix in ridge regression. Namely, A is the kernel

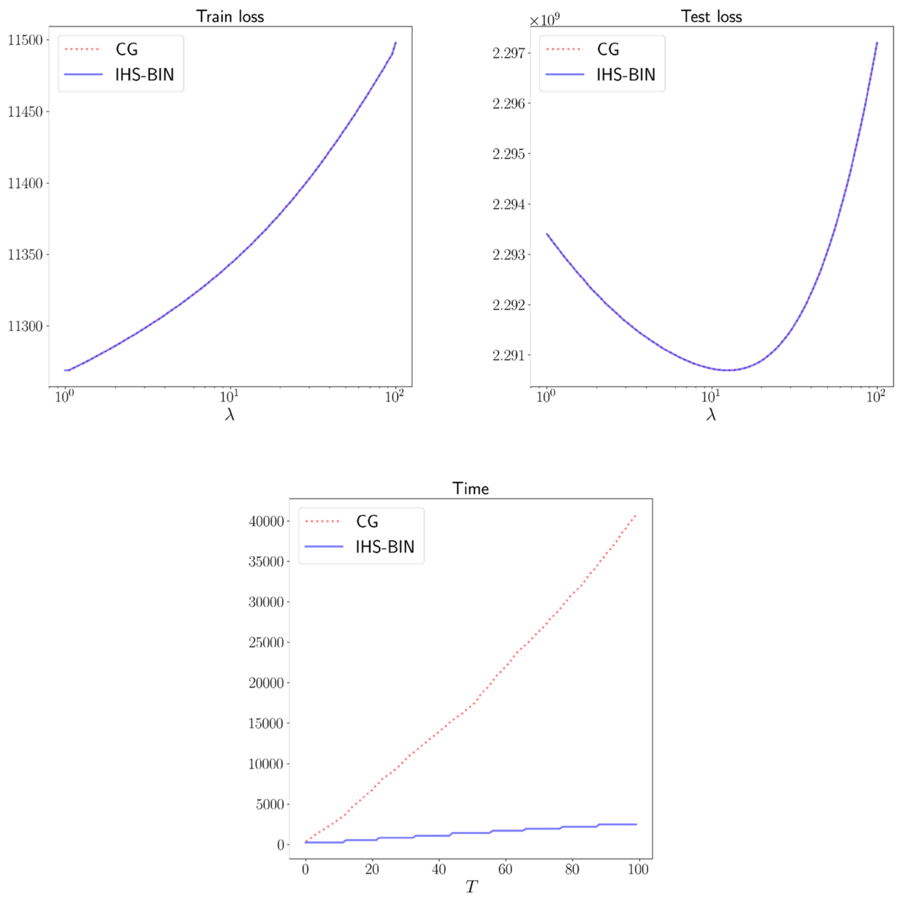

Fig. 4 Training loss, test loss and time. Avazu's click-through prediction (avazu). $n = 200000, d = 50000, m = 10000$. $\lambda_{\min} = 1$. $\lambda_{\max} = 100$. We do not calculate the eigenvalues of $A^T A$ since d is large

matrix formulated by the training data and $\tilde{A}$ is the kernel matrix formulated by the training data and test data. In short, we have

$$A_{i,j} = k(f_i, f_j), \tilde{A}_{i,j} = k(\tilde{f}_i, f_j),$$

where $k(x, y) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a positive definite kernel function, f_i is the feature vector of i-th training sample, and $\tilde{f}_j$ is the feature vector of j-th test sample. Here, we use an isotropic Gaussian kernel function

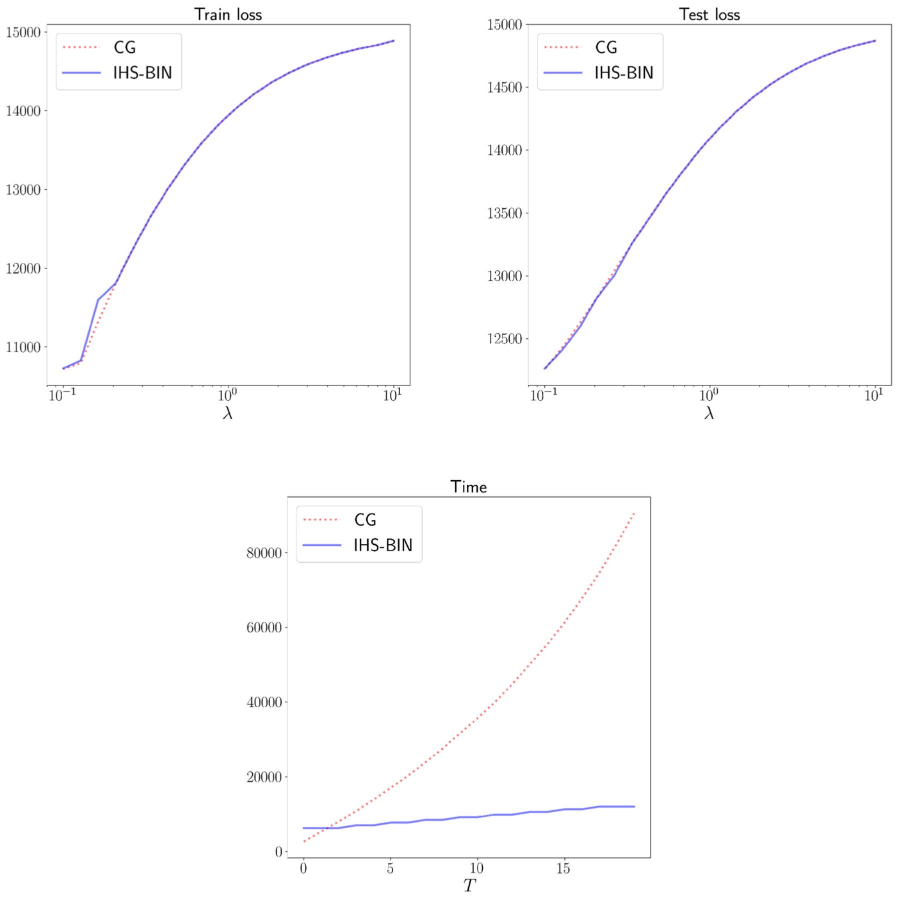

Fig. 5 Training loss, test loss and time. MNIST with quadratic feature embedding. $n = 30000, d = 608400, m = 10000$. $\lambda_{\min} = 0.1$. $\lambda_{\max} = 10$. We do not calculate the eigenvalues of AA^T since n is large

$$k(x, y) = (2\pi h)^{-d/2} \exp\left(-\frac{1}{2h}\|x - y\|_2^2\right),$$

with bandwidth $h = 1000$.

6.2 Over-determined case

We present numerical results on randomly generated data and real data in Figs. 1, 3 and 4. In Fig. 2, we present results on matrix-valued ridge regression with kernel matrix. For the problem with medium-size d (randomly generated data), we plot the eigenvalues of A^TA. The curves of train loss and test loss from IHS-BIN overlap

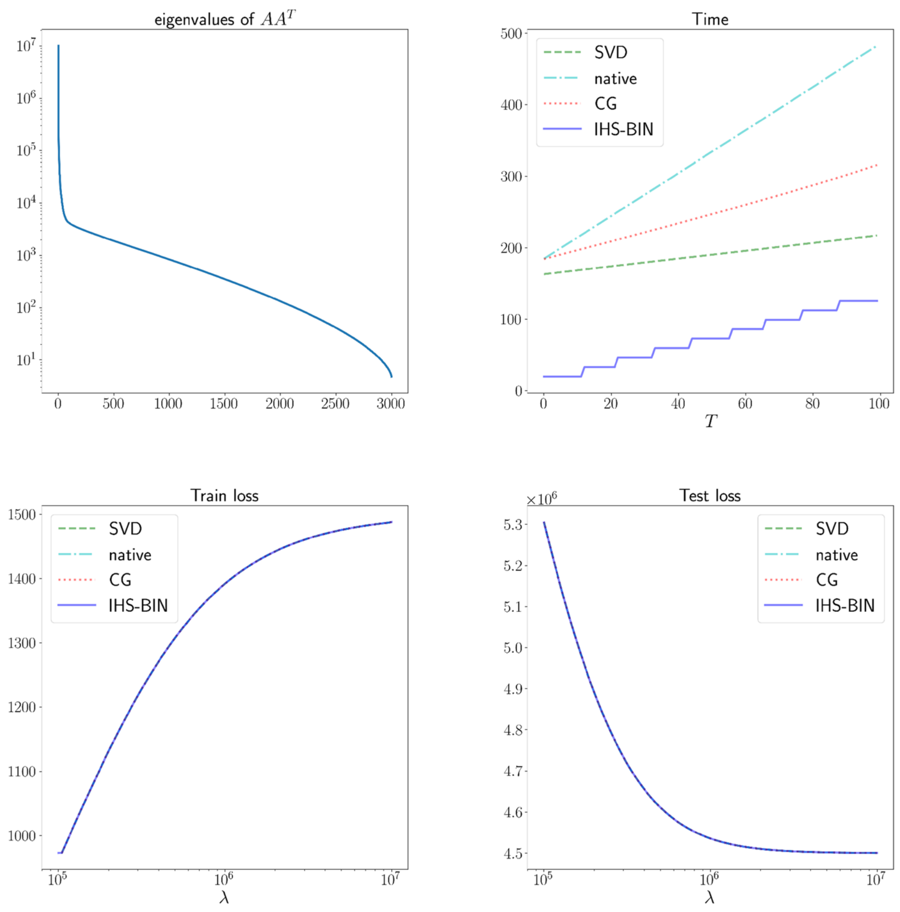

Fig. 6 Training loss, test loss and time. NIPS 2003 feature selection challenge (gisette). $n = 3000, d = 5000$, $m = 800$, $\lambda_{\min} = 10^5$. $\lambda_{\max} = 10^7$

with curves from other solvers. This indicates that IHS-BIN yields correct solutions along the regularization path. For medium-scale problems like randomly generated data and real-sim, CG and IHS-BIN outperform SVD and native. For the large-scale problem avazu and matrix-valued ridge regression, IHS-BIN can be significantly faster than CG.

6.3 Under-determined case

We also perform numerical comparisons on under-determined data matrices. We present numerical results in Figs. 5, 6, 7, 8 and 9. For the problems with medium-size n (randomly generated data, gisette, RCV1, tifdf), we plot the eigenvalues of AA^T. Similar to the over-determined case, IHS-BIN is faster than other compared

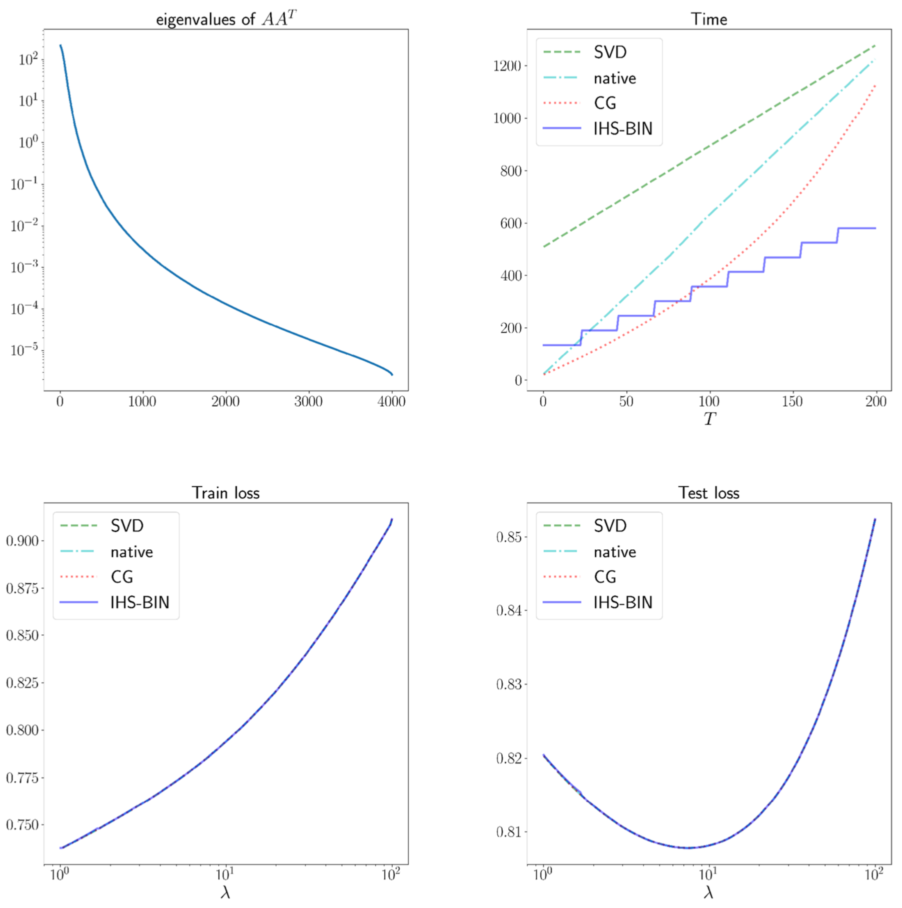

Fig. 7 Training loss, test loss and time. Randomly generated data. $n = 4000, d = 20000,\ m = 2400$. $\sigma = 0.02$. $\lambda_{\min} = 1$. $\lambda_{\max} = 100$

methods, especially for large-scale examples. The curves of train loss and test loss from IHS-BIN overlap with curves from other solvers, which implies the accuracy of IHS-BIN.

6.4 Robustness evaluation on synthetic data

We further compare our algorithm with Ridge-Sketch Python package introduced in [18] on synthetic data with different spectral properties. For IHS-BIN and Ridge-Sketch, we run for five independent trials and plot the standard deviation as shaded area. In Figs. 10 and 11, we present the performance of compared algorithms on datasets with decaying spectral and clustered spectral, respectively. We note that our proposed method is significantly faster than Ridge-Sketch. Also, IHS-BIN achieves consistently good performance on different trials. As the spectral distribution is clustered

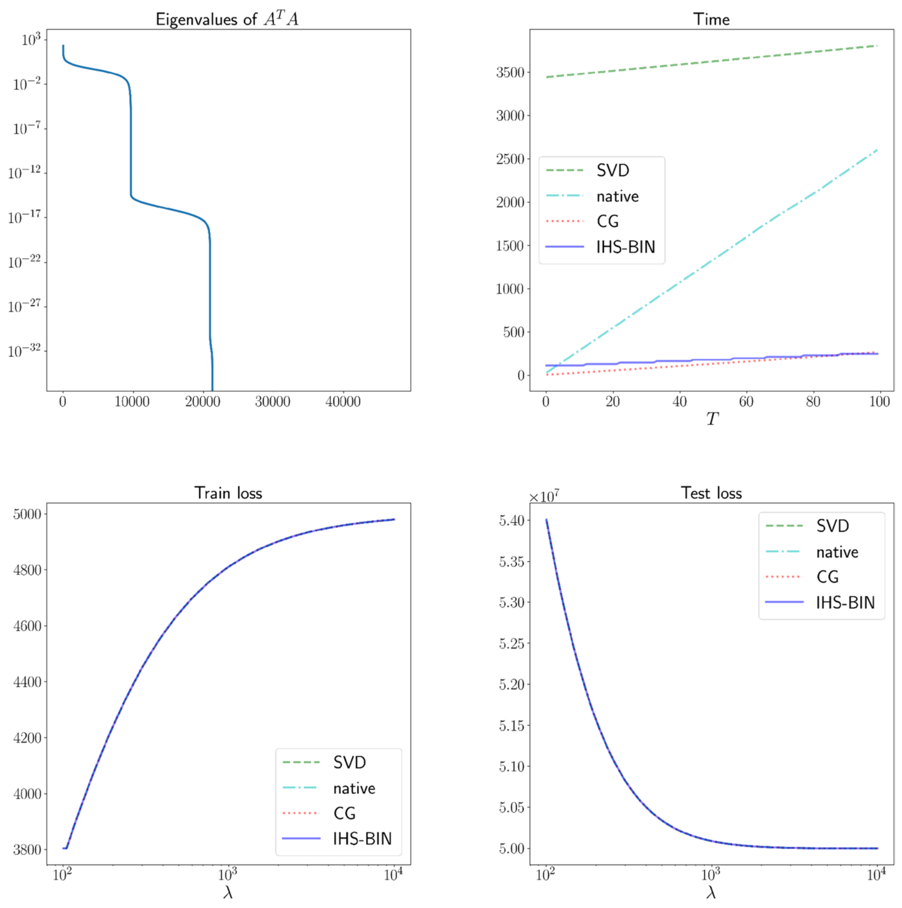

Fig. 8 Training loss, test loss and time. RCV1: A new benchmark collection for text categorization research. $n = 10000, d = 47236, m = 3000, \sigma = 0.03$. $\lambda_{\min} = 100$. $\lambda_{\max} = 10^4$

in Fig. 11, CG is slightly faster than our proposed IHS-BIN. This is expected since it is well-known that CG is ideal on data with clustered spectral distributions. However, our method is still competitive with CG even in this case. We emphasize that in all other datasets we considered in this section with the exception of the clustered data in Fig. 11, CG performs significantly worse than our method.

7 Conclusion

We presented IHS-BIN for rapidly computing the entire ridge regularization path. The algorithm is based on analyzing the gradient descent regularization path and accelerating convergence via randomized sketching. Our method improves the state-of-the-art computational complexity of obtaining the solution of ridge

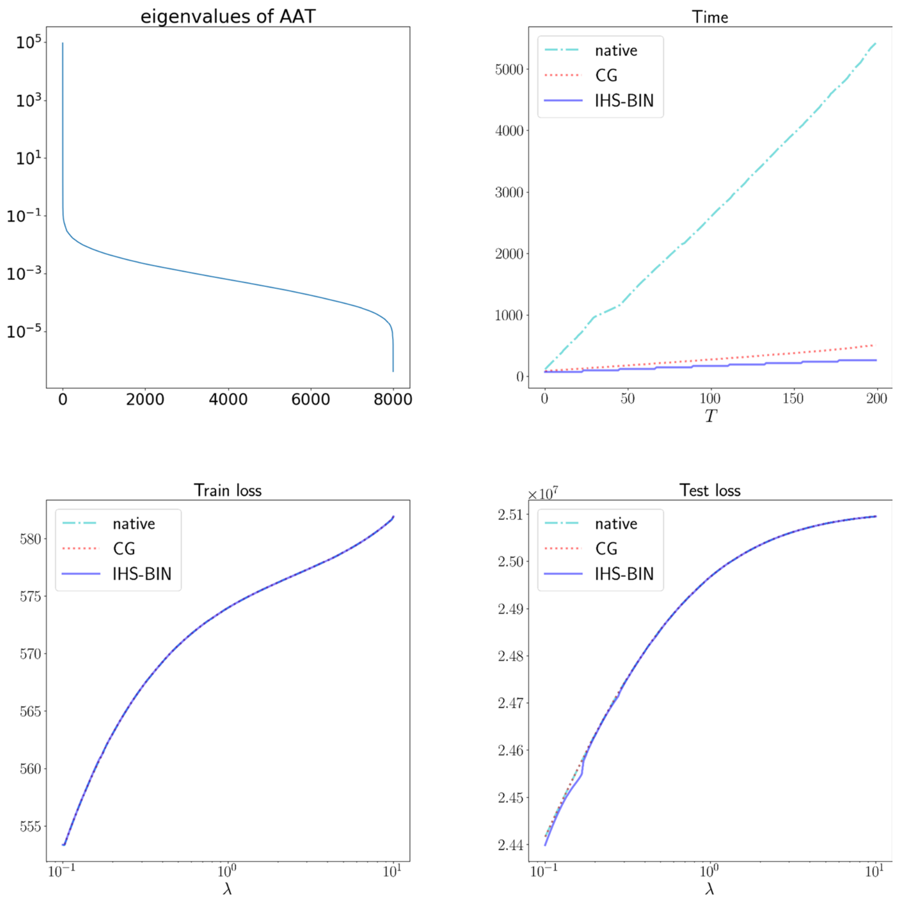

Fig. 9 Training loss, test loss and time. 10-K Corpus (E2006-tfidf). $n = 8000, d = 150360, m = 2000$. $\lambda_{\min} = 0.1$. $\lambda_{\max} = 10$

regression for T values of the regularization parameter from $\mathcal{O}(Td^2)$ or $\mathcal{O}(T\text{nnz}(A))$ to $\mathcal{O}(Td)$, when T is large. The numerical experiments also demonstrate that IHS-BIN is significantly faster than other solvers, especially for large-scale problems. Our method also leverages the low effective dimensionality of real datasets, which can be used to reduce the sketching dimension. We also investigated adaptively picking the sketch dimension based on the progress of the algorithm. We believe that our algorithm will be quite effective in automatically tuning the regularization parameters of linear models. Moreover, our method can be used

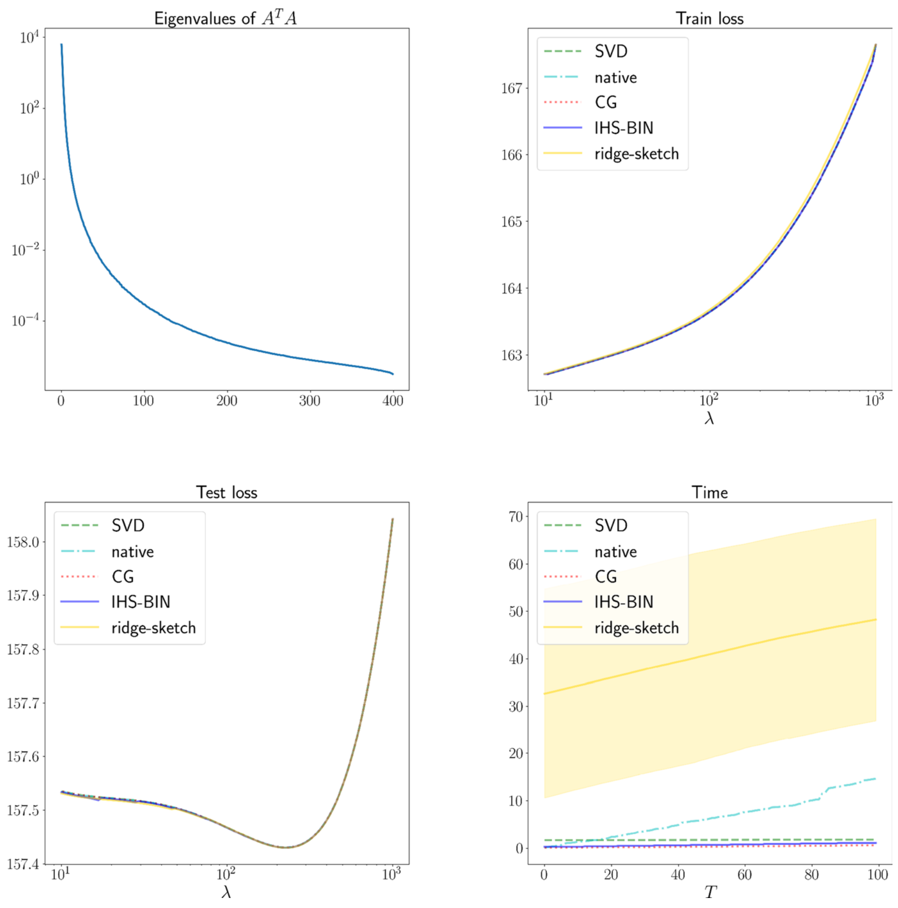

Fig. 10 Training loss, test loss and time. Randomly generated data. $n = 2000, d = 400, m = 160, \sigma = 0.4$. $\lambda_{\min} = 10$. $\lambda_{\max} = 1000$

in transfer learning and deep feature embedding for training a final linear layer and tuning the regularization parameter efficiently. One potential limitation of the proposed approach is that for medium-size data matrices with high effective dimensions, direct methods such as SVD can be more effective.

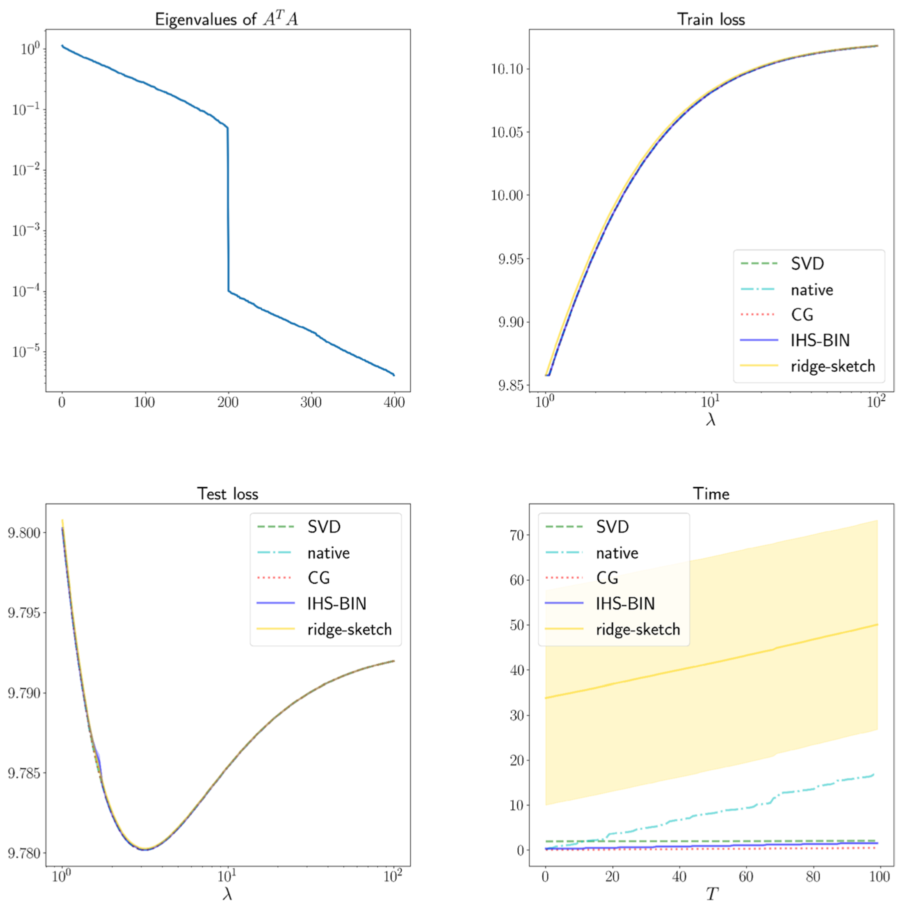

Fig. 11 Training loss, test loss and time. Randomly generated data with clustered spectral distribution. $n = 2000, d = 400, m = 160, \sigma = 0.1$. $\lambda_{\min} = 1$. $\lambda_{\max} = 100$

Acknowledgements This work was partially supported by the National Science Foundation (NSF) under grants ECCS- 2037304, DMS-2134248, NSF CAREER award CCF-2236829, the U.S. Army Research Office Early Career Award W911NF-21-1-0242, Stanford Precourt Institute, and the ACCESS—AI Chip Center for Emerging Smart Systems—sponsored by InnoHK funding, Hong Kong SAR.

Author Contributions Yifei Wang developed the theory and algorithms, performed the computations and took the lead in writing the manuscript. Mert Pilanci conceived the presented idea, developed a preliminary theory, helped in writing the manuscript and supervised the project.

Data Availability The data used in this manuscript consist of publicly available standard benchmark datasets.

Declarations

Conflict of interest All authors declare that they have no conflicts of interest.

Ethical Approval Not applicable.

References

1. Golub GH, Heath M, Wahba G (1979) Generalized cross-validation as a method for choosing a good ridge parameter. Technometrics 21(2):215–223
2. Li K-C (1985) From Stein's unbiased risk estimates to the method of generalized cross validation. Ann Statist 1:1352–1377
3. Vogel CR (2002) Computational methods for inverse problems
4. Exterkate P (2013) Model selection in kernel ridge regression. Computat Statist Data Anal 68:1–16
5. Tan C, Sun F, Kong T, Zhang W, Yang C, Liu C (2018) A survey on deep transfer learning. In: International Conference on Artificial Neural Networks. Springer pp 270–279
6. Pilanci M, Wainwright MJ (2016) Iterative hessian sketch: fast and accurate solution approximation for constrained least-squares. J Mach Learn Res 17(1):1842–1879
7. Rokem A, Kay K (2020) Fractional ridge regression: a fast, interpretable reparameterization of ridge regression. GigaScience 9(12):133
8. Rudi A, Camoriano R, Rosasco L (2015) Less is More: Nyström computational regularization. In: NIPS, pp. 1657–1665
9. Hackbusch W (1994) Iterative solution of large sparse systems of equations 95
10. Rokhlin V, Tygert M (2008) A fast randomized algorithm for overdetermined linear least-squares regression. Proc Natl Acad Sci 105(36):13212–13217
11. Woodruff DP (2014) Sketching as a tool for numerical linear algebra. arXiv preprint arXiv:1411.4357
12. Kane DM, Nelson J (2014) Sparser johnson-lindenstrauss transforms. J ACM (JACM) 61(1):1–23
13. Ailon N, 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, pp. 557–563
14. Ruhe A (1984) Rational krylov sequence methods for eigenvalue computation. Linear Algebra Appl 58:391–405
15. Lacotte J, Pilanci M (2020) Effective dimension adaptive sketching methods for faster regularized least-squares optimization. Adv Neural Inform Process Syst 33:19377–19387
16. Ozaslan IK, Pilanci M, Arikan O (2020) Regularized momentum iterative hessian sketch for large scale linear system of equations. International Conferene on Acoustics, Speech and Signal Processing
17. Avron H, Clarkson KL, Woodruff DP (2017) Sharper bounds for regularized data fitting. In: Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2017). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik
18. Gazagnadou N, Ibrahim M, Gower RM (2022) Ridgesketch: a fast sketching based solver for large scale ridge regression. SIAM J Matrix Anal Appl 43(3):1440–1468