Faster Ridge Regression via the Subsampled Randomized Hadamard Transform

Authored by:

Paramveer Dhillon
Dean P. Foster
Yichao Lu
Lyle Ungar

Abstract

We propose a fast algorithm for ridge regression when the number of features is much larger than the number of observations ($p gg n$). The standard way to solve ridge regression in this setting works in the dual space and gives a running time of $O(n2p)$. Our algorithm (SRHT-DRR) runs in time $O(nplog(n))$ and works by preconditioning the design matrix by a Randomized Walsh-Hadamard Transform with a subsequent subsampling of features. We provide risk bounds for our SRHT-DRR algorithm in the fixed design setting and show experimental results on synthetic and real datasets.

1 Paper Body

Ridge Regression, which penalizes the ‘2 norm of the weight vector and shrinks it towards zero, is the most widely used penalized regression method. It is of particular interest in the p ¿ n case (p is the number of features and n is the number of observations), as the standard ordinary least squares regression (OLS) breaks in this setting. This setting is even more relevant in today?s age of ?Big Data?, where it is common to have p n. Thus efficient algorithms to solve ridge regression are highly desirable. The current method of choice for efficiently solving RR is [19], which works in the dual space and has a running time of O(n2 p), which can be slow for huge p. As the runtime suggests, the bottleneck is the computation of XX¿ where X is the design matrix. An obvious way to speed up the algorithm is to subsample the columns of X. For example, suppose X has rank k, if we randomly subsample psubs of the p (k ¡ psubs p) features, then the matrix multiplication can be performed in O(n2 psubs) time, which is very fast! However, this speed-up comes with a big caveat. If all the signal in the problem were to be carried in just one of the p features, and if we missed this feature while sampling, we would miss all the signal. A parallel and recently popular line of research for solving large scale regression involves using some kind of random projections, for instance, transforming the data with a randomized Hadamard transform [1] or Fourier transform and then uniformly sampling observations from the resulting transformed matrix and estimating OLS on this smaller data set. The intuition behind this approach is that these frequency domain transformations uniformlize the data and smear the signal across all the observations so that there are no longer any high leverage points whose omission could unduly influence the parameter estimates. Hence, a uniform sampling in this transformed space suffices. This approach can also be viewed as preconditioning the design matrix with a carefully constructed data-independent random matrix. This transformation followed by subsampling has been used in a variety of variations, including Subsampled Randomized Hadamard Transform (SRHT) [4, 6] and Subsampled Randomized Fourier Transform (SRFT) [22, 17]. 1

In this paper, we build on the above line of research and provide a fast algorithm for ridge regression (RR) which applies a Randomized Hadamard transform to the columns of the X matrix and then samples psubs = O(n) columns. This allows the bottleneck matrix multiplication in the dual RR to be computed in O(np log(n)) time, so we call our algorithm Subsampled Randomized Hadamard Transform-Dual Ridge Regression (SRHT-DRR). In addition to being computationally efficient, we q also prove that in the fixed design setting SRHT-

DRR only increases the risk by a factor of (1 + C w.r.t. the true RR solution.

1.1

k psubs)

(where k is the rank of the data matrix)

Related Work

Using randomized algorithms to handle large matrices is an active area of research, and has been used in a variety of setups. Most of these algorithms involve a step that randomly projects the original large matrix down to lower dimensions [9, 16, 8]. [14] uses a matrix of i.i.d Gaussian elements to construct a preconditioner for least square which makes the problem well conditioned. However, computing a random projection is still expensive as it requires multiplying a huge data matrix by another random dense matrix. [18] introduced the idea of using structured random projection for making matrix multiplication substantially faster. Recently, several randomized algorithms have been developed for kernel approximation. [3] provided a fast method for low rank kernel approximation by randomly selecting q samples to construct a rank q approximation of the original kernel matrix. Their approximation can reduce the cost to O(nq 2). [15] introduced a random sampling scheme to approximate symmetric kernels and [12] accelerates [15] by applying Hadamard Walsh transform. Although our paper and these papers can all be understood from a kernel approximation point of view, we are working in the p n 1 case while they focus on large n. Also, it is worth distinguishing our setup from standard kernel learning. Kernel methods enable the learning models to take into account a much richer feature space than the original space and at the same time compute the inner product in these high dimensional space efficiently. In our p n 1 setup, we already have a rich enough feature space and it suffices to consider the linear kernel XX¿ 1 . Therefore, in this paper we propose a randomized scheme to reduce the dimension of X and accelerate the computation of XX¿ .

2

Faster Ridge Regression via SRHT

In this section we firstly review the traditional solution of solving RR in the dual and it?s computational cost. Then we introduce our algorithm SRHT-DRR for faster estimation of RR. 2.1

Ridge Regression

Let X be the n ? p design matrix containing n i.i.d. samples from the p dimensional independent variable (a.k.a. ?covariates? or ?predictors?) X such that p n. Y is the real valued n ? 1 response vector which contains n corresponding values of the dependent variable Y . is the n ? 1 homoskedastic noise vector with common variance ? 2 . Let ??? be the solution of the RR problem, i.e. 1 ??? = arg min kY ? X?k2 + ?k?k2 (1) ??p?1 n The solution to Equation (1) is ??? = (X¿ X + n?Ip)?1 X¿ Y . The step that dominates the computational cost is the matrix inversion which takes O(p3) flops and will be extremely slow when p n 1. A straight forward improvement to this is to solve the Equation (1) in the dual space. By change of variables ? = X¿ ? where ? ? n ? 1 and further letting K = XX¿ the optimization problem becomes 1 ? ? ? = arg min kY ? K?k2 + ??¿ K? (2) ??n?1 n 1

For this reason, it is standard in natural language processing applications to just use linear kernels.

2

and the solution is ? ? ? = (K + n?In)?1 Y which directly gives ??? = X¿ ? ? ? . Please see [19] for a detailed derivation of this dual solution. In the p n case the step that dominates computational cost in the dual solution is computing the linear kernel matrix K = XX¿ which takes O(n2 p) flops. This is regarded as the computational cost of the true RR solution in our setup. Since our algorithm SRHT-DRR uses Subsampled Randomized Hadamard Transform (SRHT), some introduction to SRHT is warranted. 2.2

Definition and Properties of SRHT

Following [20], for p = 2q where q is any positive integer, a SRHT can be defined as a psubs ? p (p ¿ psubs) matrix of the form: r p ?= RHD psubs where ? R is a random psubs ? p matrix the rows of which are psubs uniform samples (without replacement) from the standard basis of Rp . ? H ? Rp?p is a normalized Walsh-Hadamard matrix. The Walsh-Hadamard

matrix of size

Hp/2 Hp/2 +1 +1 with H2 = . p ? p is defined recursively: Hp = Hp/2 ?Hp/2 +1 ?1 H = ?1p Hp is a rescaled version of Hp . ? D is a p ? p diagonal matrix and the diagonal elements are i.i.d. Rademacher random variables. There are two key features that makes SRHT a nice candidate for accelerating RR when p n. Firstly, due to the recursive structure of the H matrix, it takes only O(p log(psubs)) FLOPS to compute ?v where v is a generic p ? 1 dense vector while for arbitrary unstructured psubs ? p dense matrix A, the cost for computing Av is O(psubs p) flops. Secondly, after projecting any matrix W

? p ? k with orthonormal columns down to low dimensions with SRHT, the columns of ?W ? psubs ? k are still about orthonormal. The following lemma characterizes this property: Lemma 1. Let W be an p ? k (p ¿ k) matrix where W¿ W = Ik . Let ? be a psubs ? p SRHT matrix where p ¿ psubs ¿ k. Then with probability at least 1 ? (? + epk), s c log(2k ¿ ?)k (3) k(?W) ?W ? Ik k2 ? psubs The bound is in terms of the spectral norm of the matrix. The proof of this lemma is in the Appendix. The tools for the random matrix theory part of the proof come from [20] and [21]. [10] also provided similar results. 2.3

The Algorithm

Our fast algorithm for SRHT-DRR is described below: SRHT-DRR Input: Dataset X ? n ? p, response Y ? n ? 1, and subsampling size psubs . Output: The weight parameter ? ? psubs ? 1. ? Compute the SRHT of the data: XH = X?¿ . ? Compute KH = XH X¿ H ? Compute ?H,? = (KH + n?In)?1 Y , which is the solution of Equation (2) obtained by replacing K with KH . ? Compute ?H,? = X¿ H ?H,?

3

Since, SRHT is only defined for p = 2q for any integer q, so, if the dimension p is not a power of 2, we can concatenate a block of zero matrix to the feature matrix X to make the dimension a power of 2. Remark 1. Let?s look at the computational cost of SRHT-DRR. Computing XH takes O(np log(psubs)) FLOPS [2, 6]. Once we have XH , computing ?H,? costs O(n2 psubs) FLOPS, with the dominating step being computing KH = XH X¿ H . So the computational cost for computing ?H,? is O(np log(psubs) + n2 psubs), compared to the true RR which costs O(n2 p). We will discuss how large psubs should be later after stating the main theorem.

3

Theory

In this section we bound the risk of SRHT-DRR and compare it with the risk of the true dual ridge estimator in fixed design setting. As earlier, let X be an arbitrary n ? p design matrix such that p n. Also, we have Y = X? + , where is the n ? 1 homoskedastic noise vector with common mean 0 and variance ? 2 . [5] and [3] did similar analysis for the risk of RR under similar fixed design setups. Firstly, we provide a corollary to Lemma 1 which will be helpful in the subsequent theory. Corollary 1. Let k be the rank of X. With probability at least 1 ? (? +

p) ek

(1 ? ?)K KH (1 + ?)K

where ? = C

q

k log(2k/?) . psubs

(4)

(as for p.s.d. matrices G L means G ? L is p.s.d.)

Proof. Let X = UDV¿ be the SVD of X where U ? n ? k, V ? p ? k has orthonormal columns and D ? k ? k is diagonal. Then KH = UD(V¿ ??V)DU¿ . Lemma 1 directly implies Ik (1 ? ?) (V¿ ??V) Ik (1 + ?) with probability at least 1 ? (? + epk). Left multiply UD and right multiply DU¿ to the above inequality complete the proof.

3.1

Risk Function for Ridge Regression

Let Z = E (Y) = X?. The risk for any prediction Y? ? n ? 1 is n1 E kY? ? Zk2 . For any n ? n positive symmetric definite matrix M, define the following risk function.

R(M) =

?2 Tr[M2 (M + n?In)?2] + n?2 Z ¿ (M + n?In)?2 Z n

(5)

Lemma 2. Under the fixed design setting, the risk for the true RR solution is R(K) and the risk for SRHT-DRR is R(KH). 4

Proof. The risk of the SRHT-DRR estimator is 1 E kKH ?H,? ? Zk2 n

= =

=

=

=

1 E kKH (KH + n?In)?1 Y ? Zk2 n 1 E kKH (KH + n?In)?1 Y ? E (KH (KH + n?In)?1 Y)k2 n 1 + kE (KH (KH + n?In)?1 Y) ? Zk2 n 1 E kKH (KH + n?In)?1 k2 n 1 + k(KH (KH + n?In)?1 Z ? Zk2 n 1 Tr[K2H (KH + n?In)?2 ¿] n 1 + Z ¿ (In ? KH (KH + n?In)?1)2 Z n ?2 Tr[K2H (KH + n?In)?2] n +n?2 Z ¿ (KH + n?In)?2 Z (6)

Note that the expectation here is only over the random noise and it is conditional on the Randomized Hadamard Transform. The calculation is the same for the ordinary estimator. In the risk function, the first term is the variance and the second term is the bias. 3.2

Risk Inflation Bound

The following theorem bounds the risk inflation of SRHT-DRR compared with the true RR solution. Theorem 1. Let k be the rank of the X matrix. With probability at least 1 ? (? + epk) R(KH) ? (1 ? ?)?2 R(K) where ? = C

q

(7)

k log(2k/?) psubs

Proof. Define = n?2 Z ¿ (M + n?In)?2 Z ?2 Tr[K2H (KH + n?In)?2] V (M) = n

B(M)

for any p.s.d matrix M ? n ? n. Therefore, R(M) = V (M) + B(M). Now, due to [3] we know that B(M) is non-increasing in M and V (M) is non-decreasing in M. When Equation(4) holds, R(KH)

= ?

V (KH) + B(KH) V ((1 + ?)K) + B((1 ? ?)K)

?

(1 + ?)2 V (K) + (1 ? ?)?2 B(K)

?

(1 ? ?)?2 (V (K) + B(K))

=

(1 ? ?)?2 R(K)

Remark 2. Theorem 1 gives us an idea of how large psubs should be. Assuming ? (the risk inflation ratio) is fixed, we get psubs = C k log(2k/?) = O(k). If we further assume that X is full rank, i.e. ?2 k = n, then, it suffices to choose psubs = O(n). Combining this with Remark 1, we can see that the cost of computing XH is O(np log(n)). Hence, under the ideal setup where p is huge so that the dominating step of SRHT-DRR is computing XH , the computational cost of SRHT-DRR O(np log(n)) FLOPS. 5

Comparison with PCA Another way to handle high dimensional features is to use PCA and run regression only on the top few principal components (this procedure is called PCR), as illustrated by [13] and many other papers. RR falls in the family of ?shrinkage? estimators as it shrinks the weight parameter towards zero. On the other hand, PCA is a ?keep-or-kill? estimator as it kills components with smaller eigenvalues. Recently, [5] have shown that the risk of PCR and RR are related and that the risk of PCR is bounded by four times the risk of RR. However, we believe that both PCR and RR are parallel approaches and one can be better than the other depending on the structure of the problem, so it is hard to compare SRHT-DRR with PCR theoretically. Moreover, PCA under our p n 1 setup is itself a non-trivial problem both statistically and computationally. Firstly, in the p n case we do not have enough samples to estimate the huge p ? p covariance matrix. Therefore the eigenvectors of the sample covariance matrix obtained by PCA maybe very different from the truth. (See [11] for a theoretical study on the consistency of the principal directions for the high p low n case.) Secondly, PCA requires one to compute an SVD of the X matrix, which is extremely slow when p n 1. An alternative is to use a randomized algorithm such as [16] or [9] to compute PCA. Again, whether randomized PCA is better than our SRHT-DRR algorithm depends on the problem. With that in mind, we compare SRHT-DRR against standard as well as Randomized PCA in our experiments section; We find that SRHT-DRR beats both of them in speed as well as accuracy.

4

Experiments

In this section we show experimental results on synthetic as well as real-world data highlighting the merits of SRHT, namely, lower computational cost compared to the true Ridge Regression (RR) solution, without any significant loss of accuracy. We also compare our approach against ?standard? PCA as well as randomized PCA [16]. In all our experiments, we choose the regularization constant ? via cross-validation on the training set. As far as PCA algorithms are concerned, we implemented standard PCA using the built in SVD function in MATLAB and for randomized PCA we used the block power iteration like approach proposed by [16]. We always achieved convergence in three power iterations of randomized PCA. 4.1

Measures of Performance

Since we know the true ? which generated the synthetic data, we report MSE/Risk for the fixed design setting (they are equivalent for squared loss) as measure of accuracy. It is computed as kY? ? X?k2 , where Y? is the prediction corresponding to different methods being compared. For real-world data we report the classification error on the test set. In order to compare the computational cost of SHRT-DRR with true RR, we need to estimate the number of FLOPS used by them. As reported by other papers, e.g. [4, 6], the theoretical cost of applying Randomized Hadamard Transform is O(np log(psubs)). However, the MATLAB implementation we used took about np log(p) FLOPS to compute XH . So, for SRHT-DRR, the total computational cost is np log(p) for getting XH and a further 2n2 psubs FLOPS to compute KH . As mentioned earlier, the true dual RR solution takes ? 2n2 p. So, in our experiments, we report relative computational cost which is computed as the ratio of the two. Relative Computational Cost =

4.2

p log(p) ? n + 2n2 psubs 2n2 p

Synthetic Data

We generated synthetic data with p = 8192 and varied the number of observations n = 20, 100, 200. We generated a n ? n matrix R ? M V N (0, I) where MVN(?, ?) is the Multivariate Normal Distribution with mean vector ?, variance-covariance matrix ? and ?j ? N (0, 1) ?j = 1, . . . , p. The final X matrix was generated by rotating R with a randomly generated n ? p rotation matrix. Finally, we generated the Ys as Y = X? + where i ? N (0, 1) ?i = 1, . . . , n. 6

2.6

True RR Solution PCA Randomized PCA

2.5

True RR Solution PCA Randomized PCA

2.4 2.2

2.2 2

True RR Solution PCA Randomized PCA

2 1.8

MSE/Risk

MSE/Risk

MSE/Risk

2

1.5

1.8 1.6 1.4

1.6 1.4 1.2

1 1.2

1

1 0.5

0.8

0.8 0.6 0.329 0.331 0.337 0.349 0.386 0.417 0.447 0.471 0.508 0.569

Relative Computational Cost

0.08

0.083 0.089 0.126 0.157 0.187 0.211 0.248 0.279 0.309

Relative Computational Cost

0.6

0.059 0.063 0.069 0.094 0.124 0.155 0.179 0.216 0.246 0.277

Relative Computational Cost

Figure 1: Left to right n=20, 100, 200. The boxplots show the median error rates for SRHT-DRR for different psubs . The solid red line is the median error rate for the true RR using all the features. The green line is the median error rate for PCR when PCA is computed by SVD in MATLAB. The black dashed line is median error rate for PCR when PCA is computed by randomized PCA. For PCA and randomized PCA, we tried keeping r PCs in the range 10 to n and finally chose the value of r which gave the minimum error on the training set. We tried 10 different values for psubs from n + 10 to 2000 . All the results were averaged over 50 random trials. The results are shown in Figure 1. There are two main things worth noticing. Firstly, in all the cases, SRHT-DRR gets very close in accuracy to the true RR with only ? 30% of its computational cost. SRHT-DRR also cost much fewer FLOPS than the Randomized PCA for our experiments. Secondly, as we mentioned earlier, RR and PCA are parallel approaches. Either one might be better than the other depending on the structure of the problem. As can be seen, for our data, RR approaches are always better than PCA based approaches. We hypothesize that PCA might perform better relative to RR for larger n. 4.3

Real world Data

We took the UCI ARCENE dataset which has 200 samples with 10000 features as our real world dataset. ARCENE is a binary classification dataset which consists of 88 cancer individuals and 112 healthy individuals (see [7] for more details about this dataset). We split the dataset into 100 training and 100 testing samples and repeated this procedure 50 times (so n = 100, p = 10000 for this dataset). For PCA and randomized PCA, we tried keeping r = 10, 20, 30, 40, 50, 60, 70, 80, 90 PCs and finally chose the value of r which gave the minimum error on the training set (r = 30). As earlier, we tried 10 different values for psubs : 150, 250, 400, 600, 800, 1000, 1200, 1600, 2000, 2500. Standard PCA is known to be slow for this size datasets, so the comparison with it is just for accuracy. Randomized PCA is fast but less accurate than standard (?true?) PCA; its computational cost for r = 30 can be approximately calculated as about 240np (see [9] for details), which in this case is roughly the same as computing XX¿ (? 2n2 p). The results are shown in Figure 2. As can be seen, SRHT-DRR comes very close in accuracy to the true RR solution with just ? 30% of its computational cost. SRHT-DRR beats PCA and Randomized PCA even more comprehensively, achieving the same or better accuracy at just ? 18% of their computational cost.

5

Conclusion

In this paper we proposed a fast algorithm, SRHT-DRR, for ridge regression in the p n 1 setting SRHT-DRR preconditions the design matrix by a Randomized Walsh-Hadamard Transform with a subsequent subsampling of features. In addition to being significantly faster than the true dual ridge regression solution, SRHT-DRR only inflates the risk w.r.t. the true solution by a small amount. Experiments on both synthetic and real data show that SRHT-DRR gives significant speeds up with only small loss of accuracy. We believe similar techniques can be developed for other statistical methods such as logistic regression. 7

True RR Solution PCA Randomized PCA

Classification Error

0.4

0.35

0.3

0.25

0.2

0.15

0.1

0.13

0.14

0.155 0.175 0.195 0.215 0.235 0.275 0.315 0.365

Relative Computational Cost

Figure 2: The boxplots show the median error rates for SRHT-DRR for different psubs . The solid red line is the median error rate for the true RR using all the features. The green line is the median error rate for PCR with top 30 PCs when PCA is computed by SVD in MATLAB. The black dashed line is the median error rate for PCR with the top 30 PCs computed by randomized PCA.

2 References

[1] Nir Ailon and Bernard Chazelle. Approximate nearest neighbors and the fast johnsonlindenstrauss transform. In STOC, pages 557?563, 2006. [2] Nir Ailon and Edo Liberty. Fast dimension reduction using rademacher series on dual bch codes. Technical report, 2007. [3] Francis Bach. Sharp analysis of low-rank kernel matrix approximations. CoRR, abs/1208.2015, 2012. [4] Christos Boutsidis and Alex Gittens. Improved matrix algorithms via the subsampled randomized hadamard transform. CoRR, abs/1204.0062, 2012. [5] Paramveer S. Dhillon, Dean P. Foster, Sham M. Kakade, and Lyle H. Ungar. A risk comparison of ordinary least squares vs ridge regression. Journal of Machine Learning Research, 14:1505? 1511, 2013. [6] Petros Drineas, Michael W. Mahoney, S. Muthukrishnan, and Tam?s Sarl?s. Faster least squares approximation. CoRR, abs/0710.1435, 2007. [7] Isabelle Guyon. Design of experiments for the nips 2003 variable selection benchmark. 2003. [8] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. SIAM Rev., 53(2):217?288, May 2011. [9] Nathan Halko, Per-Gunnar Martinsson, Yoel Shkolnisky, and Mark Tygert. An algorithm for the principal component analysis of large data sets. SIAM J. Scientific Computing, 33(5):2580? 2594, 2011. [10] Daniel Hsu, Sham M. Kakade, and Tong Zhang. Analysis of a randomized approximation scheme for matrix multiplication. CoRR, abs/1211.5414, 2012. [11] S. Jung

and J.S. Marron. PCA consistency in high dimension, low sample size context. Annals of Statistics, 37:4104?4130, 2009. [12] Quoc Le, Tamas Sarlos, and Alex Smola. Fastfood -approximating kernel expansions in loglinear time. ICML, 2013. [13] W.F. Massy. Principal components regression in exploratory statistical research. Journal of the American Statistical Association, 60:234?256, 1965. [14] Xiangrui Meng, Michael A. Saunders, and Michael W. Mahoney. Lsrn: A parallel iterative solver for strongly over- or under-determined systems. CoRR, abs/1109.5981, 2011. 8

[15] Ali Rahimi and Ben Recht. Random features for large-scale kernel machines. In In Neural Infomration Processing Systems, 2007. [16] Vladimir Rokhlin, Arthur Szlam, and Mark Tygert. A randomized algorithm for principal component analysis. SIAM J. Matrix Analysis Applications, 31(3):1100?1124, 2009. [17] Vladimir Rokhlin and Mark Tygert. A fast randomized algorithm for overdetermined linear least-squares regression. Proceedings of the National Academy of Sciences, 105(36):13212? 13217, September 2008. [18] Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In In Proc. 47th Annu. IEEE Sympos. Found. Comput. Sci, pages 143?152. IEEE Computer Society, 2006. [19] G. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In Proc. 15th International Conf. on Machine Learning, pages 515?521. Morgan Kaufmann, San Francisco, CA, 1998. [20] Joel A. Tropp. Improved analysis of the subsampled randomized hadamard transform. CoRR, abs/1011.1595, 2010. [21] Joel A. Tropp. User-friendly tail bounds for sums of random matrices. Foundations of Computational Mathematics, 12(4):389?434, 2012. [22] Mark Tygert. A fast algorithm for computing minimal-norm solutions to underdetermined systems of linear equations. CoRR, abs/0905.4745, 2009.

9