CS 15-859 CC: Algorithms for Big Data

Fall 2022 • Prof. David Woodruff

Notes by Fan Pu Zeng

Contents

Lecture 1 — September 8, 2022 **2**
1 Overview . . . 2
2 Regression Analysis . . . 2
3 Singular Value Decomposition . . . 4
4 Sketching to Solve Least-Squares Regression . . . 6

Lecture 2 — September 15, 2022 **7**
1 Recap of the ε-Net Argument . . . 7
2 Choosing the Sketching Matrix . . . 7
3 Faster Subspace Embeddings . . . 9

Lecture 3 — September 22, 2022 **11**
1 Recap . . . 11
2 From Vectors to Matrices (Approximate Matrix Product) . . . 11
3 Summary . . . 12

Lecture 4 — September 29, 2022 **13**
1 Affine Embeddings . . . 13
2 Low Rank Approximation . . . 13
3 Part II: High Precision Regression . . . 18

Lecture 6 — October 13, 2022 **20**
1 Recap: Row-Partition Protocol . . . 20
2 Coordinator . . . 20
3 Constructing a Coreset . . . 20

Lecture 7 — October 27, 2022 **21**
1 L_1 Regression . . . 21

CS 15-859 CC: Algorithms for Big Data Fall 2022

Lecture 1 — September 8, 2022

Prof. David Woodruff *Scribe: Fan Pu Zeng*

1 Overview

This course is about algorithms for massive data sets. We want algorithms that run in linear time or less, usually at the cost of randomization and approximation.

2 Regression Analysis

Linear regression studies linear dependencies between variables.

The standard setting is:

- a measured variable b;
- predictor variables $a_1, \ldots, a_d$;
- the assumption that
$$b = x_0 + a_1 x_1 + \cdots + a_d x_d + \varepsilon,$$
where the x_i are the model parameters and ε is noise.

We can assume $x_0 = 0$ without loss of generality, by creating a new predictor variable a_0 that is always equal to 1.

We take n measurements of b.

2.1 Matrix form

The input is an $n \times d$ matrix A and a vector $b = (b_1, \ldots, b_n)$. Here d is the number of predictor variables, and row A_i holds the predictor values for the i-th measurement.

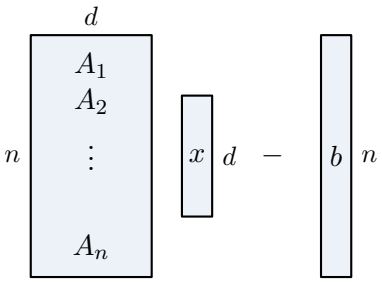

Figure 1: The regression problem in matrix form: we want Ax to be close to b.

The output is a vector x^* such that Ax^* and b are close.

Consider the *over-constrained* case, $n \gg d$. Typically there is no exact solution to $Ax = b$.

2.2 Least-squares method

Find x^* that minimizes

$$\min_{x \in \mathbb{R}^d} \|Ax - b\|_2^2 \;=\; \sum_{i=1}^{n} \big(\langle A_i, x\rangle - b_i\big)^2.$$

When the noise ε is Gaussian, x^* is the maximum likelihood estimator.

2.3 Geometry of regression

Writing $A_{*,i}$ for the i-th column of A,

$$Ax = \sum_{i=1}^{d} A_{*,i}\, x_i = \begin{bmatrix} A_{*,1} & \cdots & A_{*,d} \end{bmatrix} x.$$

So Ax is a linear combination of the d columns of A, and ranges over a d-dimensional subspace. We are choosing a vector in the span of the d columns.

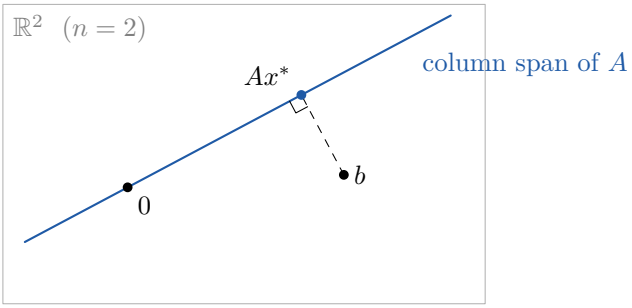

Figure 2: For $n = 2$: Ax^* is the point of the column span of A nearest to b.

Solving least squares is equivalent to computing the point of the column space of A nearest to b in ℓ_2-norm. That is, Ax^* is the *projection of b onto the column span of A*.

2.4 Normal equations

Write $b = Ax' + b'$, where b' is orthogonal to the columns of A. Then

$$\begin{aligned} \|Ax - b\|_2^2 &= \|Ax - Ax' - b'\|_2^2 \\ &= \|A(x - x') - b'\|_2^2 \\ &= \|A(x - x')\|_2^2 + \|b'\|_2^2, \end{aligned}$$

where the last step uses that b' is orthogonal to $A(x - x')$.

So x is an optimal solution if and only if

$$A^T(Ax - b) = A^T(Ax - Ax') = 0,$$

which holds by setting $x = x'$.

Normal equations. An optimal x satisfies

$$A^T Ax = A^T b.$$

If A has full column rank, then $A^T A$ has full rank, so

$$x = (A^T A)^{-1} A^T b.$$

Suppose instead that the columns of A are not linearly independent. Then the Moore–Penrose pseudoinverse gives a minimum-norm solution x.

3 Singular Value Decomposition

Any $n \times d$ matrix A can be decomposed as $A = U\Sigma V^T$:

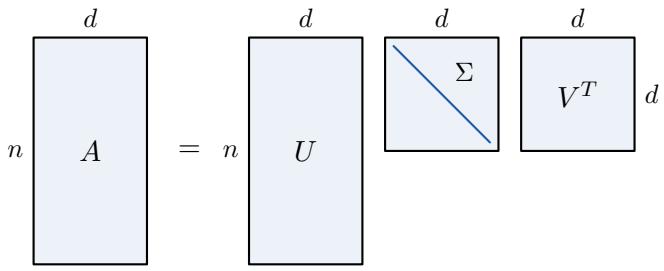

Figure 3: The (thin) SVD of an $n \times d$ matrix.

- The columns of U are orthonormal.
- Σ is a non-negative diagonal matrix with $\Sigma_{11} \geq \Sigma_{22} \geq \cdots \geq \Sigma_{dd}$.
- The columns of V are orthonormal.

Definition (Pseudoinverse)**.** The *pseudoinverse* of A is

$$A^- = V\Sigma^{-1}U^T,$$

where Σ^{-1} is the diagonal matrix with $\Sigma_{ii}^{-1} = 1/\Sigma_{ii}$ if $\Sigma_{ii} > 0$, and 0 otherwise.

If the entries of Σ are all non-zero, then

$$A^- A = I, \qquad \text{but } AA^- \text{ is not the identity!}$$

Example 1. Let $A = e_1 \in \mathbb{R}^{n \times 1}$, with SVD

$$A = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \cdot \underbrace{1}_{\Sigma} \cdot \underbrace{1}_{V^T}.$$

Then

$$AA^- = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} = \begin{bmatrix} 1 & & & \\ & 0 & & \\ & & \ddots & \\ & & & 0 \end{bmatrix},$$

which is called a *projection*.

The column span of U is the same as the column span of A.

3.1 Solving the normal equations with the pseudoinverse

Recall the normal equations $A^T Ax = A^T b$. Using

$$A^T = V\Sigma U^T, \qquad A = U\Sigma V^T, \qquad A^- = V\Sigma^{-1}U^T,$$

we can expand out

$$A^T AA^- b = V\Sigma U^T U\Sigma V^T V\Sigma^{-1}U^T b = V\Sigma\Sigma\Sigma^{-1}U^T b = V\Sigma U^T b = A^T b,$$

which is the right-hand side. So $x = A^- b$ is an optimal solution.

Any optimal solution x has the form

$$x = A^- b + (I - V'V'^T)z$$

for some vector z. Here V^T is split into blocks of rows,

$$V^T = \left[\frac{V'^T}{V''^T}\right],$$

where the columns of V' correspond to the non-zero singular values and the columns of V'' to the zero ones.

Why is this an optimal solution? We need to show

$$A^T A(I - V'V'^T)z = 0.$$

We have

$$A^T A(I - V'V'^T)z = A^T U\Sigma V^T \underbrace{(I - V'V'^T)}_{\text{projection}} z.$$

Decompose $z = z_{V'} + z_{V''}$ with $z_{V'} \perp z_{V''}$. Then $(I - V'V'^T)z = z_{V''}$, and $\Sigma V^T z_{V''} = 0$ because $z_{V''}$ lies in the span of the columns of V with zero singular values. So the expression is 0.

3.2 Time complexity

It is not known how to find the pseudoinverse without computing the SVD. Computing the SVD takes

- $O(nd^2)$ time naively;
- $O(nd^{1.376})$ time using fast matrix multiplication.

4 Sketching to Solve Least-Squares Regression

How can we find an approximate solution faster?

Goal: output x' such that, with high probability,

$$\|Ax' - b\|_2 \leq (1+\varepsilon) \min_x \|Ax - b\|_2 .$$

4.1 The sketch-and-solve paradigm

Subspace embeddings

Let $k = O(d/\varepsilon^2)$, and let S be a $k \times n$ matrix of i.i.d. normal $N(0, 1/k)$ random variables.

Problem. For a fixed x we have

$$\mathbb{P}\Big[\|SAx\|_2^2 \in (1 \pm \varepsilon)\|Ax\|_2^2\Big] \geq 1 - 2^{-\Theta(d)}.$$

We want this to hold for all x simultaneously.

Nets

Consider the sphere S^{d-1}.

Definition (γ-net)**.** A set $N \subseteq S^{d-1}$ is a *γ-net* if for every $x \in S^{d-1}$ there exists $y \in N$ such that $\|x - y\|_2 \leq \gamma$.

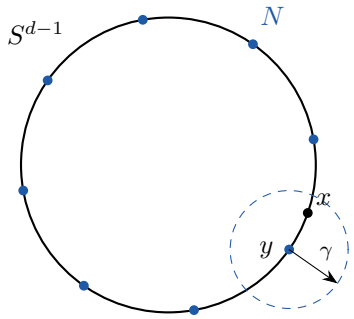

Figure 4: A γ-net N on the sphere: every point x is within distance γ of some net point y.

CS 15-859 CC: Algorithms for Big Data Fall 2022

Lecture 2 — September 15, 2022

Prof. David Woodruff *Scribe: Fan Pu Zeng*

1 Recap of the ε-Net Argument

Consider the subspace L spanned by the columns of A together with b. The net argument shows that

$$\forall y \in L, \qquad \|Sy\|_2 = (1 \pm \varepsilon) \|y\|_2 .$$

So the cost of the minimum solution is also preserved, up to a factor of $(1 \pm \varepsilon)$.

2 Choosing the Sketching Matrix

How do we choose the right sketching matrix S?

2.1 Subsampled Randomized Hadamard Transform

Let S be a *Subsampled Randomized Hadamard Transform*:

$$S = P \cdot H \cdot D,$$

where

- D is an $n \times n$ diagonal matrix with $+1$ or -1 on the diagonal: each diagonal entry is $+1$ with probability $\frac{1}{2}$ and -1 with probability $\frac{1}{2}$;
- H is the $n \times n$ Hadamard matrix, with entries $\pm\frac{1}{\sqrt{n}}$;
- P chooses a random subset of the rows of $H \cdot D$.

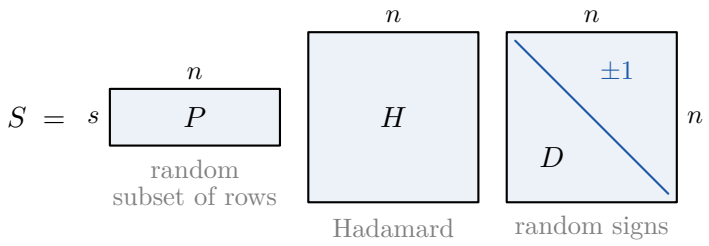

Figure 5: The Subsampled Randomized Hadamard Transform $S = PHD$.

2.2 The Hadamard matrix

Let $n = 2^k$, and index the rows and columns of H by $i, j \in \{0,1\}^k$. Then

$$H_{i,j} = \frac{(-1)^{\langle i,j \rangle}}{\sqrt{n}}.$$

The Hadamard matrix has a recursive structure. For $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ with $x_1, x_2 \in \mathbb{R}^{n/2}$,

$$H_n x = \frac{1}{\sqrt{2}} \begin{bmatrix} H_{n/2} & H_{n/2} \\ H_{n/2} & -H_{n/2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

So we can compute $H_n x$ by computing $H_{n/2} x_1$ and $H_{n/2} x_2$ recursively, and then combining them with $O(n)$ additions and subtractions. The running time satisfies

$$T(n) = 2\,T(n/2) + O(n), \qquad \text{so} \qquad T(n) = O(n \log n).$$

Claim 1. The columns (and rows) of H are orthonormal.

Proof. For columns j, j',

$$\sum_i H_{i,j} H_{i,j'} = \frac{1}{n} \sum_i (-1)^{\langle i,j \rangle} (-1)^{\langle i,j' \rangle} = \frac{1}{n} \sum_i (-1)^{\langle i, j+j' \rangle},$$

where $j + j'$ is taken mod 2. If $j = j'$, then $j + j' = 0$ and the sum is $\frac{1}{n} \cdot n = 1$. If $j \neq j'$, then $j + j'$ is non-zero, so $\langle i, j + j' \rangle$ is even for exactly half of the $i \in \{0,1\}^k$ and odd for the other half, and the sum is 0. ∎

2.3 Why the SRHT is a subspace embedding

$S \cdot A$ can be computed in $O(nd \log n)$ time, by applying the fast Hadamard transform to each of the d columns of DA.

Assume without loss of generality that A has orthonormal columns, so that $\|Ax\|_2 = 1$ for every unit vector x. It suffices to show that

$$\|SAx\|_2^2 = \|PHDAx\|_2^2 = 1 \pm \varepsilon \qquad \text{for all unit } x.$$

HD is a rotation matrix, so

$$\|HDAx\|_2^2 = \|Ax\|_2^2 = 1 \qquad \text{for all unit } x.$$

HD is like a random rotation, but it is not truly random, since a truly random rotation would be too expensive.

Let $y = Ax$.

Lemma 1 (Flattening lemma)**.** *For any fixed y,*

$$\mathbb{P}\left[\|HDy\|_\infty \geq C\, \frac{\sqrt{\log(nd/\delta)}}{\sqrt{n}} \right] \leq \frac{\delta}{2d}.$$

Here $\|HDy\|_\infty$ is the largest absolute value of a coordinate of HDy.

2.4 Operator norm

Definition (Operator norm)**.** For a matrix X with SVD $X = U\Sigma V^T$, the *operator norm* is

$$\|X\|_2 = \sigma_1(X) = \max_y \frac{\|Xy\|_2}{\|y\|_2},$$

where $\sigma_1(X)$ is the largest singular value of X.

Write $U = HDA$, which has orthonormal columns. The sampling matrix P is an $s \times n$ matrix with a single non-zero entry $\sqrt{n}/\sqrt{s}$ in each row. A non-zero entry in the i-th column samples the i-th row of U.

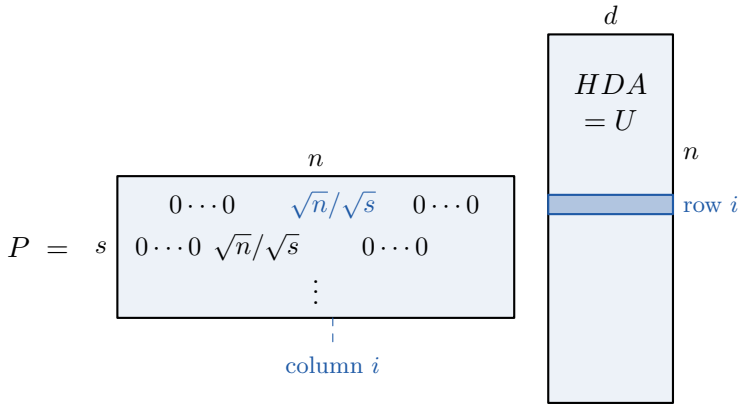

Figure 6: Each row of P samples one row of $U = HDA$: a non-zero entry in column i picks out row i of U.

Norms must satisfy the triangle inequality. The operator norm satisfies the triangle inequality:

$$\|X + Y\|_2 \leq \|X\|_2 + \|Y\|_2 .$$

Key idea: we can conclude the bound for all x if we bound the operator norm.

2.5 Running time

The total running time is

$$\underbrace{O(nd \log n)}_{\text{compute } SA} + \underbrace{\text{poly}\left(\frac{d \log n}{\varepsilon}\right)}_{\text{regression}}.$$

This is much better than the nd^2 for the Gaussian sketch, and it is nearly optimal in the matrix dimensions when $n \gg d$.

3 Faster Subspace Embeddings

What if the matrix is sparse? We use the *CountSketch* matrix.

All sketching matrices are short and fat:

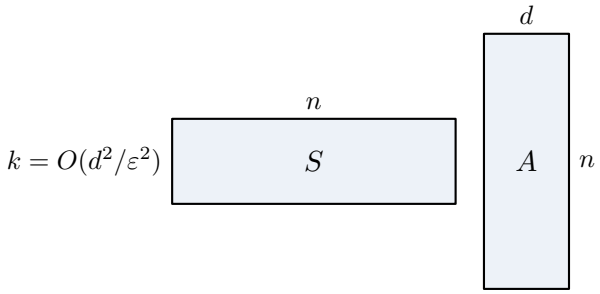

Figure 7: The sketching matrix S is short and fat: it has $k = O(d^2/\varepsilon^2)$ rows, while A has n rows.

Suppose A is also sparse, and is given as a list of its non-zero entries. Let $\text{nnz}(A)$ be the number of non-zero entries of A. Then we can compute SA in $O(\text{nnz}(A))$ time.

In real life, matrices are sparse!

CS 15-859 CC: Algorithms for Big Data Fall 2022

Lecture 3 — September 22, 2022

Prof. David Woodruff *Scribe: Fan Pu Zeng*

1 Recap

Last time we saw faster subspace embeddings, using the CountSketch matrix.

The JL property implies approximate matrix product.

2 From Vectors to Matrices (Approximate Matrix Product)

Minkowski's inequality

For vectors x, y,

$$\|x + y\|_p \leq \|x\|_p + \|y\|_p .$$

Hölder's inequality

Recall the Cauchy–Schwarz inequality. There is no reason to stop at the ℓ_2 norm! Let p, q satisfy

$$\frac{1}{p} + \frac{1}{q} = 1.$$

Then

$$\langle x, y \rangle = \sum_i x_i y_i \leq \Big(\sum_i |x_i|^p\Big)^{1/p} \Big(\sum_i |y_i|^q\Big)^{1/q} = \|x\|_p \|y\|_q .$$

Cauchy–Schwarz is the case $p = q = 2$.

CountSketch satisfies the JL property

CountSketch satisfies the JL property. Recall that the CountSketch matrix S is built from a hash function $h : [n] \to [k]$ and random signs $\sigma : [n] \to \{-1, +1\}$: column i of S has a single non-zero entry $\sigma(i)$, in row $h(i)$.

The functions h and σ do not need to be truly random. It is enough for h to be 2-wise independent and for σ to be 4-wise independent.

Definition (2-wise independence)**.** h is drawn from a *2-wise independent family* if for all $i \neq j$ and all $k_1, k_2 \in \{1, \ldots, k\}$,

$$\mathbb{P}[h(i) = k_1 \text{ and } h(j) = k_2] = \frac{1}{k^2}.$$

Such a function can be represented with $O(\log n)$ bits!

3 Summary

With probability at least 9/10,

$$\|S(AX - B)\|_F^2 = (1 \pm \varepsilon) \|AX - B\|_F^2$$

for all X simultaneously, if S has $O(d^2/\varepsilon^2)$ rows. The row count comes from three properties that S must satisfy; next to each is the number of rows it needs:

(1) subspace embedding: d^2/ε^2 rows;

(2) approximate matrix product: d/ε^2 rows;

(3) $\|SB^*\|_F^2 = (1 \pm \varepsilon) \|B^*\|_F^2$: $1/\varepsilon^2$ rows, where $B^* = AX^* - B$ is the residual of an optimal solution X^*.

The number of rows of S does not depend on the number of columns of B! B can even have infinitely many columns.

Next: low rank approximation.

CS 15-859 CC: Algorithms for Big Data Fall 2022

Lecture 4 — September 29, 2022

Prof. David Woodruff *Scribe: Fan Pu Zeng*

1 Affine Embeddings

Last time we introduced affine embeddings.

Recap. Now X and B are matrices, and B has a large number of columns. We want to show that

$$\|SAX - SB\|_F = (1 \pm \varepsilon) \|AX - B\|_F \qquad \text{for all } X.$$

2 Low Rank Approximation

Today we want to apply affine embeddings to low rank approximation.

We could use the SVD and zero out all but the top k singular values, but this is too slow. Instead, we relax the problem: find A' such that

$$\|A - A'\|_F \leq (1 + \varepsilon) \|A - A_k\|_F ,$$

where A' is an $n \times d$ matrix of rank k. Such an A' can be written as a product $A' = LR$:

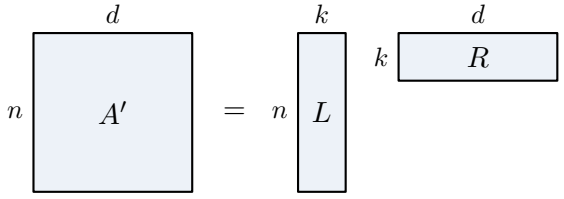

Figure 8: A rank-k matrix A' written as the product of an $n \times k$ matrix L and a $k \times d$ matrix R.

This has only $nk + dk$ parameters. We write A' in terms of the matrix product instead of writing out the whole thing.

2.1 Solution to low rank approximation

Idea: compute SA, where the number of rows of S is small (k/ε).

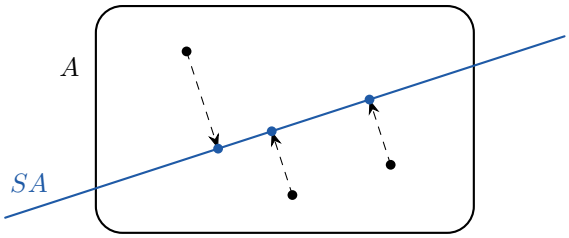

Figure 9: The rows of A as points, each projected onto the row span of SA.

SA takes a small number of linear combinations of the rows of A:

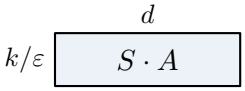

Figure 10: The sketch SA has k/ε rows and d columns.

The row span of SA is at most k/ε-dimensional, and lies in $\mathbb{R}^d$.

What if we take all these points and project each one onto the closest point in the row span of SA? Now we have n points in the row space of SA, but they are k/ε-dimensional.

Recall that the SVD takes nd^2 time. Think of A as n points, each living in d dimensions. We want instead n points in k/ε dimensions.

Idea: we will work with a basis of SA, and represent the n points using this basis.

2.2 What can S be?

- Gaussian.
- SRHT / Fast Johnson–Lindenstrauss.
- CountSketch ($(k^2+k)/\varepsilon$).

It turns out all of these matrices work!

2.3 Why? A connection to regression

Consider

$$\min_X \|A_k X - A\|_F ,$$

where A_k is the best rank-k approximation to A. (We don't know what A_k is, else we would be done!)

A_k has rank k, so $A_k X$ has rank at most k. Then the best rank-k solution for X is just the identity.

Let S be an affine embedding. Then

$$\|SA_k X - SA\|_F = (1 \pm \varepsilon) \|A_k X - A\|_F \qquad \text{for all } X.$$

(The number of rows of S depends only on the rank of A_k.)

We want to solve this sketched regression problem:

$$\operatorname*{argmin}_X \|SA_kX - SA\|_F = (SA_k)^- SA.$$

Decompose it column by column, and apply the normal equations. Then

$$\left\|A_k(SA_k)^- SA - A\right\|_F \leq (1+\varepsilon)\left\|A_k - A\right\|_F.$$

This shows that there exists a good projection matrix in SA! Since $A_k(SA_k)^-$ has rank at most k,

$$\min_{\text{rank-}k\ X} \|XSA - A\|_F^2 \leq \left\|A_k(SA_k)^- SA - A\right\|_F^2 \leq (1+\varepsilon)\left\|A - A_k\right\|_F^2.$$

Why is this different from normal regression? The rank of the solution is constrained. So the above says that if we do this, we have a $(1+\varepsilon)$-approximation!

2.4 Computing the best rank-k approximation in the row span of SA

If we do not constrain X, the normal equations give, by the Pythagorean theorem,

$$\|XSA - A\|_F^2 = \left\|XSA - A(SA)^- SA\right\|_F^2 + \left\|A(SA)^- SA - A\right\|_F^2.$$

Recall that B^-B is the projection onto the row span of B. So $A_i(SA)^-SA$ is the row A_i projected onto the row span of SA.

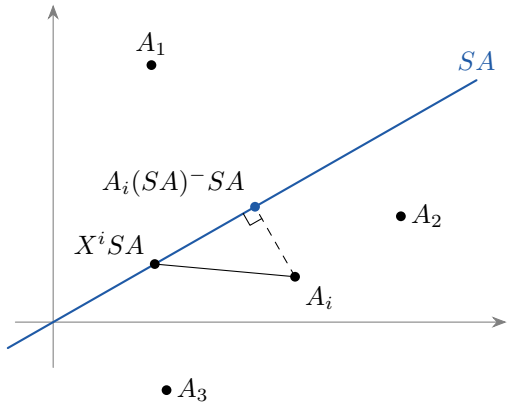

Figure 11: Row A_i, its projection $A_i(SA)^-SA$ onto the row span of SA, and another point X^iSA of the row span; the three form a right triangle.

We want

$$\|XSA - A\|_F^2 = \sum_i \left\|X^iSA - A_i\right\|_2^2,$$

where X^i is the i-th row of X. Regression gives

$$\min_{X^i} \left\|A_i - X^iSA\right\|_2^2,$$

which is finding the closest point by regression. Everything is just transposed from before: the optimal X^i satisfies $X^iSA = A_i(SA)^-SA$.

Hence we want to minimize $\|XSA - A\|_F^2$ over rank-k matrices X. By the Pythagorean identity above, the second term, $\|A(SA)^-SA - A\|_F^2$, does not depend on X. So it suffices to minimize the first term, $\|XSA - A(SA)^-SA\|_F^2$. Assume the rows of SA have full rank. (We can write $SA = U\Sigma V^T$ in its thin SVD, which means throwing out linearly dependent rows.) Then

$$\begin{aligned}\min_{\text{rank-}k\ X} \|XSA - A(SA)^-SA\|_F^2 &= \min_{\text{rank-}k\ X} \|XU\Sigma - A(SA)^-U\Sigma\|_F^2 && (V^T \text{ is orthonormal})\\ &= \min_{\text{rank-}k\ Y} \|Y - A(SA)^-U\Sigma\|_F^2 && (U\Sigma \text{ has full rank}).\end{aligned}$$

So now we can just compute the SVD of $A(SA)^-U\Sigma$! Its rows are only k/ε-dimensional, since $U\Sigma$ has only k/ε columns.

2.5 The bottleneck

But even though we could take the SVD of $A(SA)^-U\Sigma$, we don't have this matrix! This is because:

- The running time of the SVD is $\min(nd^2, n^2d)$.
- Computing SA and $(SA)^-$ is fine.
- $U\Sigma$ comes from the SVD of SA.
- But $A((SA)^-U\Sigma)$ cannot be computed efficiently: we have no idea how dense or sparse $(SA)^-U\Sigma$ is.

Current algorithm.

1. Compute SA. This takes $\text{nnz}(A)$ time.
2. Project each row of A onto the row span of SA. (This seems hard.)
3. Find the best rank-k approximation of the projected points inside the row space of SA. (We can just do an SVD.)

The bottleneck is step 2. How can we fix this?

But projection is regression! So why don't we approximate the projection instead?

2.6 Sketching on the right

We approximate

$$\min_{\text{rank-}k\ X} \|X(SA) - A\|_F^2 .$$

We can sketch on the left, so why not sketch on the right? This is precisely an affine embedding! So we can sketch on the right and preserve norms for all X. That is, with R an affine embedding, we solve

$$\min_{\text{rank-}k\ X} \|X(SA)R - AR\|_F^2 .$$

It remains to solve this problem.

But first: how long does it take to compute?

- AR: $\operatorname{nnz}(A)$ time.
- SAR: $\operatorname{nnz}(SA) \le \operatorname{nnz}(A)$ time.

So we can write down the problem in $\operatorname{nnz}(A)$ time. We want $\operatorname{nnz}(A) + (n+d)\operatorname{poly}(k/\varepsilon)$ time overall.

Now we can just output

$$\operatorname*{argmin}_{\text{rank-}k\ X} \|XSA - A\|_F^2 .$$

By the Pythagorean theorem,

$$\min_{\text{rank-}k\ X} \|XSAR - AR\|_F^2 = \left\|AR(SAR)^- SAR - AR\right\|_F^2 + \min_{\text{rank-}k\ X} \left\|XSAR - AR(SAR)^- SAR\right\|_F^2 .$$

Compute

$$\min_{\text{rank-}k\ Y} \left\|Y - AR(SAR)^-(SAR)\right\|_F^2$$

with the SVD; using Y is just a change of variables. Now we can compute the SVD, since the sketched dimensions are only $\operatorname{poly}(k/\varepsilon) \times \operatorname{poly}(k/\varepsilon)$. This means we can do the SVD very fast! The SVD takes $n \cdot \operatorname{poly}(k/\varepsilon)$ time, and we can solve for Y.

Remark 1. A question in lecture: recall that S had to be an affine embedding for A_k. So S must be a subspace embedding of A_k. So we can see that S has a bit larger rank than k; that is, SAR usually has rank greater than k.

2.7 Recovering X

So now we have Y. But we don't want Y, we want X.

Claim 2. We can just do a change of variables.

Why is this okay? Because the optimal Y looks like $XSAR$; that is, the rows of Y are in the row span of SAR. (If Y is not in the row span of SAR, we can just get rid of those components and make the objective smaller!)

So $Y = XSAR$ for some X. Then $Y(SAR)^- SA = XSA$ is our solution. Done! We output it in factored form, because there is no time to output the whole matrix.

This works because SAR has more columns than rows, since it needs to be a subspace embedding:

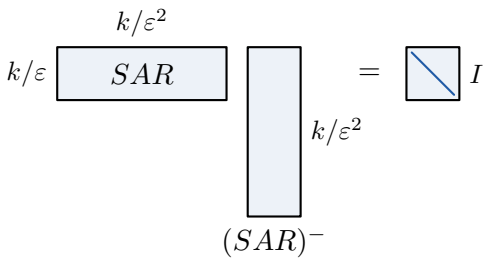

Figure 12: The wide matrix SAR times its pseudoinverse is the identity.

That is, multiplying by the pseudoinverse on the left gives a projection, but on the right it gives the identity.

2.8 Outputting the factorization

$Y(SAR)^- SA$ has rank k. As in Figure 8, we output L (an $n \times k$ matrix) and R (a $k \times d$ matrix). So how do we get L and R? The factors have sizes

$$\underbrace{Y}_{n \times \mathrm{poly}(k/\varepsilon)} \quad \underbrace{(SAR)^-}_{\mathrm{poly}(k/\varepsilon) \times \mathrm{poly}(k/\varepsilon)} \quad \underbrace{SA}_{\mathrm{poly}(k/\varepsilon) \times d},$$

and Y has rank k. We can write $Y = U\Sigma V^T$, keeping only the non-zero singular values, so U is $n \times k$ and ΣV^T has k rows. Then $L = U$, and the other factors go into R: $R = \Sigma V^T (SAR)^- SA$, which is $k \times d$. (Here R is the right factor, not the sketching matrix.) We are done!

2.9 Recap of low rank approximation

1. Instead of the SVD, compute SA.
2. Compute SAR and AR (R is also a CountSketch). Steps 1 and 2 take $\mathrm{nnz}(A)$ time.
3. Set up the optimization problem
$$\min_{\text{rank-}k\ Y} \|Y - AR(SAR)^-(SAR)\|_F^2$$
and solve it with the SVD. This takes $n \cdot \mathrm{poly}(k/\varepsilon)$ time.
4. Output $Y(SAR)^- SA$ in factored form. This takes $(n+d)\,\mathrm{poly}(k/\varepsilon)$ time.

The overall time is
$$\mathrm{nnz}(A) + (n+d)\,\mathrm{poly}(k/\varepsilon).$$
So we really only needed that S is an affine embedding.

3 Part II: High Precision Regression

Here we use sketching to help gradient descent.

In some applications you want ε to be very tiny. So sometimes we want running time $\mathrm{poly}(d)\log(1/\varepsilon)$.

Definition (Condition number)**.** The *condition number* of A is
$$\kappa_A = \frac{\sup_{\|x\|_2=1} \|Ax\|_2}{\inf_{\|x\|_2=1} \|Ax\|_2} = \frac{\text{max singular value}}{\text{min singular value}}.$$

The smallest possible condition number is 1. We want this to be a constant! The time complexity of a lot of algorithms depends on $\kappa(A)$. We use sketching to reduce $\kappa(A)$ to $O(1)$.

3.1 Small QR decomposition

Write

$$SA = QR^{-1}, \qquad \text{so} \qquad SAR = Q.$$

Then R is a good preconditioner of SA.

Definition (Preconditioner)**.** Let B be an $n \times d$ matrix and R a $d \times d$ matrix. R is a *good preconditioner* if $\kappa(BR) = O(1)$ and $\text{colspan}(BR) = \text{colspan}(B)$.

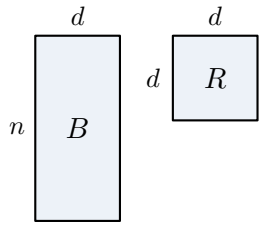

Figure 13: Preconditioning: an $n \times d$ matrix B is multiplied on the right by a $d \times d$ matrix R.

The condition number

$$\kappa(B) = \frac{\sigma_1(B)}{\sigma_d(B)}$$

could be large.

Next: leverage score sampling.

CS 15-859 CC: Algorithms for Big Data Fall 2022

Lecture 6 — October 13, 2022

Prof. David Woodruff *Scribe: Fan Pu Zeng*

1 Recap: Row-Partition Protocol

KVW: decrease the communication overhead by sketching.

The row span of SA is a good space to project to.

2 Coordinator

Matrices: S, T_1, T_2.

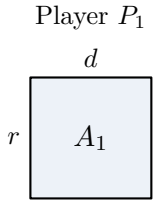

Figure 14: Player P_1 holds the matrix A_1, with r rows and d columns.

3 Constructing a Coreset

Take $m = k + k/\varepsilon$ singular values and the corresponding rows of V^T.

CS 15-859 CC: Algorithms for Big Data Fall 2022

Lecture 7 — October 27, 2022

Prof. David Woodruff *Scribe: Fan Pu Zeng*

1 L_1 Regression

1.1 Net for the ℓ_1-ball

For $y \in \mathbb{R}^n$,

$$\|y\|_2 \leq \|y\|_1 \leq \sqrt{n}\, \|y\|_2 .$$

1.2 Cauchy random variables

A Cauchy random variable has a heavy tail: its density decays like $1/z^2$. It has undefined expectation and infinite variance.