Low Rank Approximation and Regression in Input Sparsity Time

David Woodruff
IBM Almaden

Joint work with Ken Clarkson (IBM Almaden)

Talk Outline

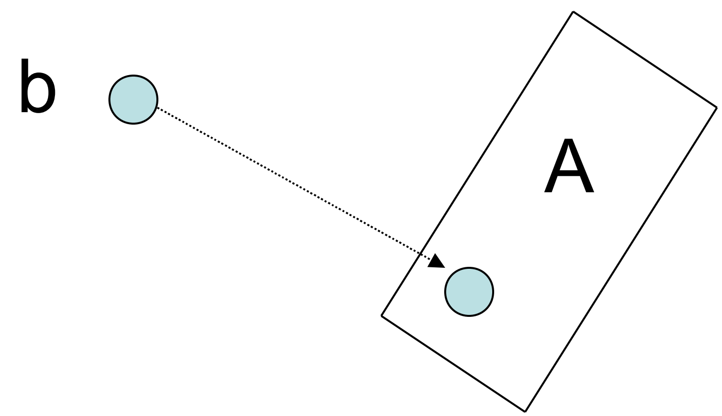
- Least-Squares Regression
  - Known Results
  - Our Results
- Low-Rank Approximation
  - Known Results
  - Our Results
- Experiments

Least-Squares Regression

- A is an n x d matrix, b an n x 1 column vector
- Consider over-constrained case, n À d
- Find x so that $|Ax-b|_2$ · $(1+\varepsilon) \min_y |Ay-b|_2$
- Allow a tiny probability of failure (depends only on randomness of algorithm, not on the input)

The Need for Approximation

- For $y = A^{-}b$, Ay is the "closest" point in the column space of A to the vector b



- Computing y exactly takes $O(nd^2)$ time
- Too slow, so we allow $\varepsilon > 0$ and a tiny probability of failure

Subspace Embeddings

- Let $k = O(d/\varepsilon^2)$
- Let S be a k x n matrix of i.i.d. normal N(0,1/k) random variables
- For any fixed d-dimensional subspace, i.e., the column space of an n x d matrix A
  - W.h.p., for all x in R^d, $|SAx|_2 = (1\pm\varepsilon)|Ax|_2$
- Entire column space of A is preserved

Why is this true?

Subspace Embeddings – A Proof

- Want to show $|SAx|_2 = (1\pm\varepsilon)|Ax|_2$ for all x
- Can assume columns of A are orthonormal (since we prove this for all x)
- By rotational invariance, SA is a k x d matrix of i.i.d. N(0,1/k) random variables
- Well-known that singular values of SA are all in the range $[1-\varepsilon, 1+\varepsilon]$
- Hence, $|SAx|_2 = (1\pm\varepsilon)|Ax|_2$

What does this have to do with regression?

Subspace Embeddings for Regression

- Want x so that $|Ax-b|_2 \cdot (1+\varepsilon) \min_y |Ay-b|_2$
- Consider subspace L spanned by columns of A together with b
- Then for all y in L, $|Sy|_2 = (1\pm \varepsilon) |y|_2$
- Hence, $|S(Ax-b)|_2 = (1\pm \varepsilon) |Ax-b|_2$ for all x
- Solve $\text{argmin}_y |(SA)y – (Sb)|_2$
- Given SA, Sb, can solve in poly(d/ε) time

But computing SA takes $O(nd^2)$ time right?

Subspace Embeddings - Generalization

- S need not be a matrix of i.i.d normals
- Instead, a "Fast Johnson-Lindenstrauss matrix" S suffices
- Usually have the form: S = P*H*D
  - D is a diagonal matrix with +1, -1 on diagonals
  - H is the Hadamard transform
  - P just chooses a random (small) subset of rows of H*D
- SA can be computed in O(nd log n) time

Previous Work vs. Our Result

- [AM, DKM, DV, …, Sarlos, DMM, DMMW, KN]
  Solve least-squares regression in
  $O(nd \log d) + \mathrm{poly}(d/\varepsilon)$ time

- Our Result
  Solve least-squares regression in
  $O(\mathrm{nnz}(A)) + \mathrm{poly}(d/\varepsilon)$ time,
  where $\mathrm{nnz}(A)$ is number of non-zero entries of A

  Much faster for sparse A, e.g., $\mathrm{nnz}(A) = O(n)$

Our Technique

- Better subspace embedding!
- Define k x n matrix S, for k = poly(d/ε)
- S is really sparse: single randomly chosen non-zero entry per column

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Surprising Part

- For certain $k = \text{poly}(d/\varepsilon)$, w.h.p., for all x,

$$|SAx|_2 = (1 \pm \varepsilon)\ |Ax|_2$$

- Since S is so sparse, SA can be computed in nnz(A) time

- Regression can be solved in nnz(A) + poly(d/ε) time

Why Did People Miss This?

- Usually put a net on a d-dimensional subspace, and argue for all z in the net,

$$|SAz|_2 = (1 \pm \varepsilon)\ |Az|_2$$

- Since the net has size exp(d), need S to preserve the lengths of exp(d) vectors

- If these vectors were arbitrary, the above S would not work!

So how could this possibly work?

Leverage Scores

- Suffices to prove for all unit vectors x

$$|SAx|_2 = (1 \pm \varepsilon)\ |Ax|_2$$

- Can assume columns of A are orthonormal
  - $|A|_F^2 = d$
- Let T be any set of size d/¯ containing all i 2 [n] for which $|A_i|_2^2$ ¸ ¯
  - T contains the large leverage scores
- For any unit x in R^d,

$$|(Ax)_i| = |<A_i, x>| \cdot |A_i|_2 ¢ |x|_2 \cdot |A_i|_2$$

- Say a coordinate i is heavy if $|(Ax)_i|$ ¸ $¯^{1/2}$
  - Heavy coordinates are a subset of T!

Perfect Hashing

- View map S as randomly hashing coordinates into k buckets, and maintaining an inner product with a sign vector in each bucket

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- If $k > 10d^2 / \bar{\ }^2 = 10\,|T|^2$, then with constant probability, all coordinates in T are perfectly hashed
- Call this event E and condition on E

The Three Error Terms

- Suppose $y = Ax$ for an x in R^d

- $y = y_T + y_{[n] \setminus T}$

- $|Sy|_2^2 = |Sy_T|_2^2 + |Sy_{[n]\setminus T}|_2^2 + 2\langle Sy_T, Sy_{[n]\setminus T}\rangle$

The Large Coordinate Error Term

- Need to bound $|Sy_T|_2^2$
- Since event E occurs, $|Sy_T|_2^2 = |y_T|_2^2$

The Small Coordinate Error Term

- Need to bound $|Sy_{[n]\setminus T}|_2^2$

- Key point: $|y_{[n]\setminus T}|_1$ is small

- [DKS]: There is an ® ¼ ε^2/d so that if $k = \Omega(\log(1/\delta)/\varepsilon^2)$ for a mapping of our form S, then for any vector y with $|y|_1$ = O(®),

  $\Pr[|Sy|_2^2 = |y_{[n]/T}|_2^2 \pm O(\varepsilon)] = 1 - O(\delta)$

- Set ¯ = O(®²) = $1/\mathrm{poly}(d/\varepsilon)$ so $|y_{[n]\setminus T}|_1$ = O(®)

- Hence, $\Pr[|Sy_{[n]\setminus T}|_2^2 = |y_{[n]/T}|_2^2 \pm O(\varepsilon)\ ] = 1 - O(\delta)$

The Cross-Coordinate Error Term

- Need to bound $|<Sy_T, Sy_{[n]\backslash T}>|$
- Sy_T only has support on $|T|$ coordinates
- Let G µ [n]\T be such that each i 2 G hashes to a bucket containing a j 2 T
- $|<Sy_T, Sy_{[n]\backslash T}>| = |<Sy_T, Sy_G>| \cdot |Sy_T|_2 ¢ |Sy_G|_2$
- $|Sy_T|_2 = |y_T|_2 \cdot 1$ by event E
- $\Pr[|Sy_G|_2 \cdot |y_G|_2 + O(\varepsilon)] = 1\text{-}O(\delta)$ by [DKS]
- $\Pr[|y_G|_2 \cdot \varepsilon] = 1\text{-}O(\delta)$ by Hoeffding

- Hence, $\Pr[|<Sy_T, Sy_{[n]\backslash T}>| \cdot 2\varepsilon] = 1\text{-} O(\delta)$

Putting it All Together

- Given that event E occurs, for any fixed y, with probability at least 1-O(δ):

$$|Sy|_2^2 = |Sy_T|_2^2 + |Sy_{[n]\setminus T}|_2^2 + 2\langle Sy_T, Sy_{[n]\setminus T}\rangle$$

$$= |y_T|_2^2 + |y_{[n]\setminus T}|_2^2 \pm O(\varepsilon)$$

$$= |y|_2^2 \pm O(\varepsilon)$$

$$= (1 \pm O(\varepsilon))|y|_2^2$$

The Net Argument

[F, M, AHK]: If for any fixed pair of unit vectors x,y, a random d x d matrix M satisfies

$$\Pr[|x^T M y| = O(\varepsilon)] > 1\text{-}\exp(-d),$$

then for every unit vector x, $|x^T M x| = O(\varepsilon)$

- We apply this to $M = (SA)^T SA - I_d$
- Set $\delta = \exp(-d)$:
  - For any x,y with probability 1-exp(-d):

$$|SA(x+y)|_2 = (1 \pm \varepsilon)|A(x+y)|_2$$

$$|SAx|_2 = (1 \pm \varepsilon)|Ax|_2 \,,$$

$$|SAy|_2 = (1 \pm \varepsilon)|Ay|_2$$

Hence, $|x^T M y| = O(\varepsilon)$

Talk Outline

- Least-Squares Regression
  - Known Results
  - Our Results
- Low-Rank Approximation
  - Known Results
  - Our Results
- Experiments

Low Rank Approximation

A is an n x n matrix

Want to output a rank k matrix A', so that

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

w.h.p., where $A_k = \text{argmin}_{\text{rank k matrices B}} |A-B|_F$

Previous results:

$$\text{nnz}(A)^*(k/\varepsilon + k \log k) + n^*\text{poly}(k/\varepsilon)$$

Our result: $\text{nnz}(A) + n^*\text{poly}(k/\varepsilon)$

Technique

- [CW] Let S be an n x k/ε^2 matrix of i.i.d. ±1 entries, and R an n x k/ε matrix of i.i.d. ±1 entries. Let $A' = AR(S^T AR)^- S^T A$.
- Can extract low rank approximation from A'

- Our result: similar analysis works if R, S are our new subspace embedding matrices
- Operations take nnz(A) + n*poly(k/ε) time

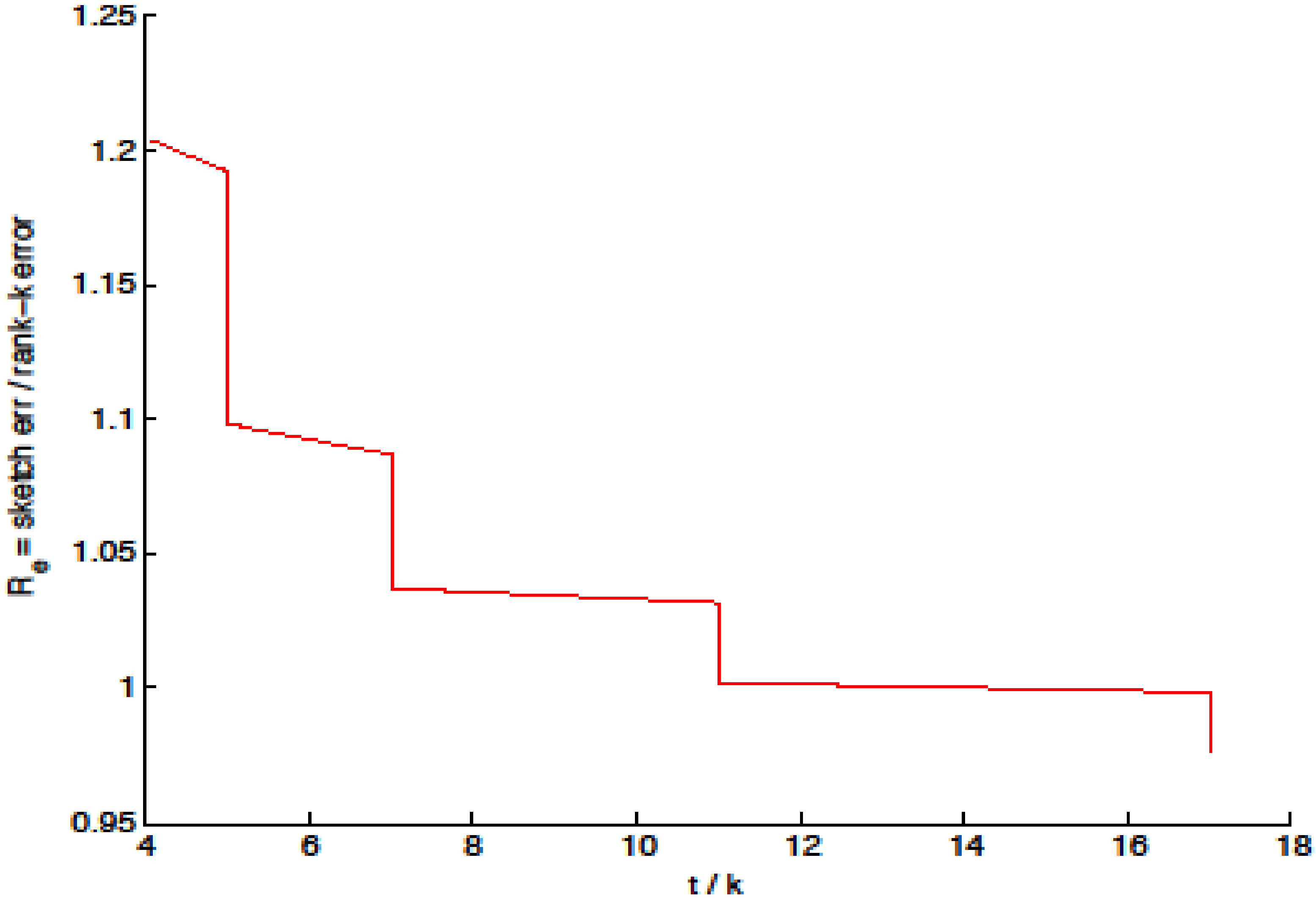
Talk Outline

- Least-Squares Regression
  - Known Results
  - Our Results
- Low-Rank Approximation
  - Known Results
  - Our Results
- Experiments

Experiments

- Looked at low rank approximation
  - n ¼ 600, nnz(A) at most 10^5
- Test matrices from University of Florida Sparse Matrix Collection
- 40 different sparsity patterns, representing different application areas
- 500 different matrices
- Dominant time is computing SA, takes same time as one matrix-vector product in Lanczos

Experiments



Conclusions

- Gave new subspace embedding of a d-dimensional subspace of R^n in time:

  $$\text{nnz}(A) + \text{poly}(d/\varepsilon)$$

- Achieved the same time for regression, improving prior nd log d time algorithms
- Achieved $\text{nnz}(A) + n^*\text{poly}(k/\varepsilon)$ time for low-rank approximation, improving previous nd log d + $n^*\text{poly}(k/\varepsilon)$ time algorithms