Topic 14: Least Squares Regression and LSH Recap

David Woodruff

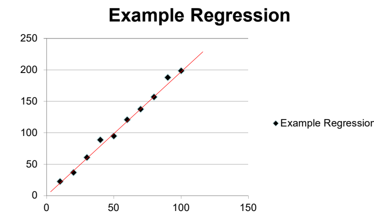
Regression

Linear Regression

- Understand linear dependencies between variables in the presence of noise.

Example

- Ohm's law V = R · I
- Find linear function that best fits the data



Talk Outline

- Least squares regression
- Sketching for least squares regression
- Locality Sensitive Hashing Recap

Regression

Standard Setting

- One measured variable b
- A set of predictor variables $a_1, \ldots, a_d$
- Assumption:

$$b = x_0 + a_1 x_1 + \ldots + a_d x_d + \varepsilon$$

ε is assumed to be noise and the x_i are model parameters we want to learn

Can assume $x_0 = 0$ by increasing d to d+1 and setting $a_0 = 1$

Now consider n observations of b

Regression

Matrix form

Input: $n \times d$-matrix A and a vector $b=(b_1,\ldots, b_n)$
n is the number of observations; d is the number of predictor variables

Output: x^* so that Ax^* and b are close

- Consider the over-constrained case, when $n \gg d$
- Note: there may not be a consistent solution x^*

Least Squares Regression

- In HW 7, you looked at

$$x^* = \mathrm{argmin}_x |Ax - b|_2^2,$$

and argued if A is n x n symmetric, then $A^2x^* = Ab$

- Extends to non-symmetric matrices: for $A \in R^{n \times d}$ and $b \in R^n$, if $x^* = \mathrm{argmin}_x |Ax - b|_2^2$, then $A^T A x^* = A^T b$
- If the columns of A are linearly independent,
  - $A^T A$ is d x d and full rank
- Closed form expression: $x^* = (A^T A)^{-1} A^T b$

Least Squares Regression

- Find x^* that minimizes $|Ax-b|_2^2$

For a vector $y \in R^n$, $|y|_2^2 = \sum_{i=1,..,n} y_i^2$

Least Squares Regression

- In practice, n is very large and d is moderate
- Computing $x^* = (A^T A)^{-1} A^T b$ takes nd^2 time
- Want running time nnz(A) + poly(d)
  - nnz(A) is the number of non-zero entries of A, and you need this time just to read the input
  - poly(d) is hopefully a low-degree polynomial in d

Talk Outline

- Least squares regression
- Sketching for least squares regression
- Locality Sensitive Hashing Recap

Sketching to Solve Least Squares Regression

- Draw S from a k x n random family of matrices, for a value k << n
  - S is known as the sketching matrix
- Compute S*A and S*b
- Output the solution x‘ to $\min_{x'} |(SA)x-(Sb)|_2$ using our closed-form expression
- Black box reduction to original, smaller problem

Sketching to Solve Least Squares Regression

- How to find an approximate solution x to $\min_x |Ax-b|_2$?
- Goal: output x‘ for which $|Ax'-b|_2 <= (1+\varepsilon) \min_x |Ax-b|_2$ with say, 99% probability
- Would like a running time of the form

$$\mathrm{nnz}(A) + \mathrm{poly}(d/\epsilon)$$

- nnz(A) is at most nd, so improves our earlier nd^2 time

Fast Sketching Matrices

- CountSketch matrix
- Define k x n matrix S, for $k = O(d^2/\varepsilon^2)$
- S is really sparse: single randomly chosen non-zero entry per column

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

Key Property: S*A computable in nnz(A) time

S*A Computable in nnz(A) Time

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

- For each column y of A, can compute S*y in nnz(y) time. Why?
- For each non-zero entry of y, it indexes into a column of S and there is a single non-zero entry in that column, so can update Sy in O(1) time per entry

Subspace Embeddings for Regression

- Want x so that $|Ax-b|_2 \leq (1+\varepsilon) \min_y |Ay-b|_2$
- 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 $\text{argmin}_y |(SA)y – (Sb)|_2$

It remains to show SA is a subspace embedding with k = $O(\frac{d^2}{\epsilon^2})$ rows

Subspace Embeddings

S is a subspace embedding if for 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 useful for regression?

Approximate Matrix Product

- Let C and D be any two matrices for which C has n columns and D has n rows
- Let S be a CountSketch matrix with n columns. Then,

$\Pr[|CS^TSD – CD|_F^2 \leq [6/(\delta(\text{\# rows of S}))]*|C|_F^2 |D|_F^2] \geq 1-\delta$, where for a matrix E, $|E|_F^2$ is the sum of squares of its entries

- Proof: variance calculation like you did in last recitation – will do it in this week's recitation ☺

Orthonormality

- For any n x d matrix A with linearly independent columns,
  - There's a d x d invertible matrix R so the columns of AR have length 1 and are perpendicular
- What is $|ARx|_2^2$ for a unit vector x?
  - $|ARx|_2^2 = \ |\sum_i (AR)_i x_i|^2$
  - $= \sum_i |(AR)_i x_i|^2 + \sum_{i \neq j} < (AR)_i x_i, (AR)_j x_j > \ = |x|_2^2$
- What is $(AR)^T AR$?

From Matrix Product to Subspace Embeddings

- Suffices to show for all unit x,
  $|x^T A^T S^T SA\,x - x^T x| \leq |A^T S^T SA - I|_F \leq \varepsilon$
- Matrix product result implies

$\Pr[|CS^TSD - CD|_F^2 \leq [6/(\delta(\text{\# rows of S}))] * |C|_F^2 |D|_F^2] \geq 1 - \delta$

- Set $C = A^T$ and $D = A$.
- Then $|A|_F^2 = d$ and (# rows of S) = $6 d^2/(\delta\varepsilon^2)$, which shows $|A^T S^T SA - I|_F \leq \varepsilon$

From Matrix Product to Subspace Embeddings

- Want: w.h.p., for all x in R^d, $|SAx|_2 = (1 \pm \varepsilon)|Ax|_2$
- Can assume columns of A are orthonormal
  - Unit length and perpendicular to each other
- Suffices to show $|SAx|_2 = 1 \pm \varepsilon$ for all unit x
  - For regression, apply S to [A, b]
- SA is a $6d^2/(\delta\varepsilon^2)$ x d matrix

From Matrix Product to Subspace Embeddings

- Still need for all unit x, $|x^T A^T S^T SA\,x - x^T x| \leq |A^T S^T SA - I|_F$
- Follows if we show $|ABC|_F \leq |A|_F |B|_F |C|_F$ for any matrices A, B, and C
- The above follows if we show $|AB|_F \leq |A|_F |B|_F$ for any two matrices A and B
- $|AB|_F^2 = \sum_{\text{rows } A_i \text{ and columns } B_j} < A_i, B_j >^2$

$$\leq \sum_{\text{rows } A_i \text{and columns } B_j} |A_i|_2^2 |B_j|_2^2 = |A|_F^2 |B|_F^2$$

Wrapup

- Goal: output x‘ for which $|Ax'-b|_2 <= (1+\varepsilon) \min_x |Ax-b|_2$ with say, 99% probability
- We used the sketch and solve paradigm to solve this in nnz(A) + poly(d/ϵ) time

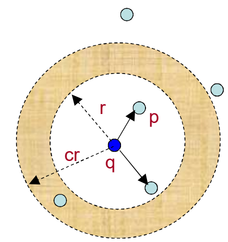
Approximate NNS

c-approximate

- r-near neighbor problem: given a new point q, report a point $p \in D$ s.t. $d(p,q) \leq r$ $\boxed{cr}$

  if there exists a point at distance $\leq r$

- Randomized: a point p returned with 90% probability



Talk Outline

- Least squares regression
- Sketching for least squares regression
- Locality Sensitive Hashing Recap

Locality Sensitive Hashing

Random hash function h on R^d satisfying:

$P_1 =$ for close pair (when “not-so-small”)
$\Pr[h(q) = h(p)]$ is “high”

$P_2 =$ for far pair (when $d(q,p) > cr$)
$\Pr[h(q) = h(p)]$ is “small”

Use several hash tables

n^ρ, where $\rho = \frac{\log 1/P_1}{\log 1/P_2}$

24

LSH for Hamming space

- Hash function g is usually a concatenation of "primitive" functions:
  - $g(p) = \langle h_1(p), h_2(p), \dots, h_k(p) \rangle$
- Fact 1: $\rho_g = \rho_h$
- Example: Hamming space $\{0,1\}^d$
  - $h(p) = p_j$, i.e., choose j^{th} bit for a random j
  - $g(p)$ chooses k bits at random
  - $\Pr[h(p) = h(q)] = 1 - \frac{\mathrm{Ham}(p,q)}{d}$
  - $P_1 = 1 - \frac{r}{d} \approx e^{-r/d}$
  - $P_2 = 1 - \frac{cr}{d} \approx e^{-c\ /d}$
  - $\rho = \frac{\log 1/P_1}{\log 1/P_2} \approx \frac{r/d}{cr/d} = \frac{1}{c}$

25

Full Algorithm

- **Data structure** is just $L = n^\rho$ hash tables:
  - Each hash table uses a fresh random function $g_i(p) = \langle h_{i,1}(p), \dots, h_{i,k}(p) \rangle$
  - Hash all dataset points into the table
- **Query**:
  - Check for collisions in each of the hash tables
  - until we encounter a point within distance cr
- **Guarantees:**
  - Space: $O(nL\log n) = O(n^{1+\rho} \log n)$ bits, plus space to store original points
  - Expected Query time: $O(L \cdot (k + d)) = O(n^\rho \cdot (k + d))$
  - 50% probability of success

26

Choice of parameters k, L ?

- L hash tables with $g(p) = \langle h_1(p), \dots, h_k(p) \rangle$
- Pr[collision of far pair] $= P_2^k$ $\boxed{\text{set } k \text{ s.t. } = 1/n}$
- Pr[collision of close pair] = $P_1^k = (P_2^\rho)^k = 1/n^\rho$
  - Success probability for a hash table: P_1^k
  - $L = O(1/P_1^k)$ tables should suffice
- Runtime as a function of P_1, P_2 ?
  - $O\left(\frac{1}{P_1^k}(\mathrm{timeToHash} + nP_2^k d)\right)$
- Hence $L = O(n^\rho)$

27

Analysis: correctness

- Let p^* be an r-near neighbor
  - If does not exist, algorithm can output anything
- Algorithm fails when:
  - near neighbor p^* is not in the searched buckets $g_1(q), g_2(q), \dots, g_L(q)$
- Probability of failure:
  - Probability q, p^* do not collide in a hash table: $\leq 1 - P_1^k$
  - Probability they do not collide in L hash tables at most

$$(1 - P_1^k)^L = \left(1 - \frac{1}{n^\rho}\right)^{n^\rho} \leq 1/e$$

28

Analysis: Runtime

- Runtime dominated by:
  - Hash function evaluation: $O(L \cdot k)$ time
  - Distance computations to points in buckets
- Distance computations:
  - Care only about far points, at distance $> cr$
  - In one hash table, we have
    - Probability a far point collides is at most $P_2^k = 1/n$
    - Expected number of far points in a bucket: $n \cdot \frac{1}{n} = 1$
  - Over L hash tables, expected number of far points is L
- Total: $O(Lk) + O(Ld) = O(n^\rho(k + d))$ in expectation

29