Lecture 23

Fast Algorithms for Least Square Regression

Course: Algorithms for Big Data

Instructor: Hossein Jowhari

Department of Computer Science and Statistics
Faculty of Mathematics
K. N. Toosi University of Technology

Spring 2021

Linear Regression

- d variables (model parameters)
- n linear equations (observations)
- $n \gg d$ (over-constrained system)

$$\left\{\begin{array}{ll}
2x_1 + 3x_4 - 6x_5 + 12x_6 & = 12 \\
x_1 - x_2 - 5x_3 + 12x_4 + x_5 & = 1 \\
x_2 + x_3 + 6x_5 & = 10 \\
2x_1 + x_2 + x_4 + x_5 & = -5 \\
-2x_1 + 3x_2 + 2x_4 - 9x_5 & = 5 \\
8x_1 - 4x_2 + x_3 + 4x_4 & = 0 \\
x_2 + 10x_3 + 5x_5 & = 12 \\
x_1 - 10x_4 - 2x_5 & = -8
\end{array}\right.$$

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \qquad \boldsymbol{A} \in \mathbb{R}^{n\times d}, \qquad \boldsymbol{b} \in \mathbb{R}^n$$

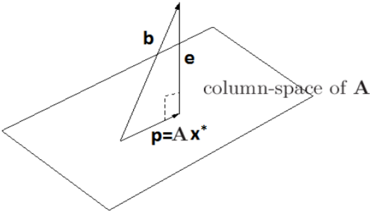
Choose $\boldsymbol{x} \in \mathbb{R}^d$ so that $\boldsymbol{A}\boldsymbol{x}$ is close to $\boldsymbol{b}$

$\boldsymbol{A}\boldsymbol{x}$ ranges over all linear combinations of d columns of $\boldsymbol{A}$

Finding a closest point in the column space of $\boldsymbol{A}$ to the vector $\boldsymbol{b}$

Least square regression

$$\arg\min_{x} \|\boldsymbol{Ax} - \boldsymbol{b}\|_2 = \sum_{i=1}^{n} (b_i - A_{i^*}.\boldsymbol{x})^2$$



$$\boldsymbol{A}^T\boldsymbol{e} = \boldsymbol{0} \Rightarrow \boldsymbol{A}^T(\boldsymbol{b} - \boldsymbol{Ax}^*) = \boldsymbol{0}$$

$$\boldsymbol{A}^T\boldsymbol{Ax}^* = \boldsymbol{A}^T\boldsymbol{b} \Rightarrow \text{normal equation}$$

If $\boldsymbol{A}$ is full-rank (has d independent columns), the unique solution is

$$\boldsymbol{x}^* = (\boldsymbol{A}^T\boldsymbol{A})^{-1}(\boldsymbol{A}^T\boldsymbol{b})$$

If $\boldsymbol{A}$ is not full-rank there are multiple solutions. One solution is

$$\boldsymbol{x}^* = \boldsymbol{A}^\dagger\boldsymbol{b}$$

Here $\boldsymbol{A}^\dagger$ is called the Moore-Penrose pseudoinverse of $\boldsymbol{A}$.

$$\boldsymbol{A}^\dagger = \boldsymbol{V}\boldsymbol{\Sigma}^\dagger\boldsymbol{U}^T, \qquad \boldsymbol{A} = \boldsymbol{U}_{n\times d}\boldsymbol{\Sigma}_{d\times d}\boldsymbol{V}_{d\times d}^T$$

Here $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ is the SVD decomposition of $\boldsymbol{A}$

Assuming n is large, finding a solution $\boldsymbol{x}^*$ takes a lot time (at least nd^2 time).

If we settle for an approximate solution, there is a faster randomized algorithm using sketching techniques.

T. Sarlós. Improved approximation algorithms for large matrices via random projections. 2006.

Find $\boldsymbol{x}$ where

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 \leq (1 + \epsilon)\|\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b}\|_2$$

Similar to what we had in JL lemma:

There is a matrix $\boldsymbol{S} \in \mathbb{R}^{r \times n}$ with random entries (where $r = \Theta(\frac{d}{\epsilon^2})$) such that with probability $1 - \exp(-d)$ for a $\boldsymbol{x} \in \mathbb{R}^d$

$$\|\boldsymbol{S}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})\|_2 \leq (1 + \epsilon)\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2$$

$$\Downarrow \text{(See the reference)}$$

We can show with probability at least $1 - 1/4$:

$$\min_{\boldsymbol{x}} \|\boldsymbol{S}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})\|_2 \leq (1 + \epsilon)\|\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b}\|_2$$

Strategy for the exact solution:

1. Output the exact solution x to the regression problem $\min_{\mathbf{x}} \| \mathbf{A}\, \mathbf{x} - \mathbf{b} \|_2$.

Time complexity: $\Omega(nd^2)$

Strategy for the approximate solution:

1. Sample a random matrix $\mathbf{S}$.
2. Compute $\mathbf{S} \cdot \mathbf{A}$ and $\mathbf{S} \cdot \mathbf{b}$.
3. Output the exact solution x to the regression problem $\min_{\mathbf{x}} \|(\mathbf{SA})\mathbf{x} - (\mathbf{Sb})\|_2$.

Sarlos in his paper shows that using special random matrices $\boldsymbol{S}$ one can obtain the matrix product $\boldsymbol{S}\boldsymbol{A}$ in time $O(nd \log d)$.

This gives the following time complexity for the approximate strategy:

Time complexity: $O(nd \log d) + \mathrm{poly}(d/\epsilon)$

Subsequently, Clarkson and Woodruff show the following improved result.

Time complexity: $nnz(\boldsymbol{A}) + \mathrm{poly}(d/\epsilon)$

Here $nnz(\boldsymbol{A})$ is the number of non-zero entries in matrix $\boldsymbol{A}$.