Computational Complexity Analysis: Least Squares vs. Gradient Descent

George Michailidis

gmichail@ucla.edu

STAT 102B

Least Squares Problem

- Given data matrix $X \in \mathbb{R}^{n \times p}$ and response vector $y \in \mathbb{R}^n$
- Objective: Find $\beta \in \mathbb{R}^p$ that minimizes the sum of squared errors (SSE)

$$\min_{\beta} \ \mathrm{SSE}(\beta) = \frac{1}{2n} \|X\beta - y\|_2^2 \tag{1}$$

- The closed form solution is given by:

$$\hat{\beta} = (X^\top X)^{-1} X^\top y \tag{2}$$

- This is exact, but can be computationally intensive for large data sets, especially in terms of number of predictors p

Clarification on Notation

In the lecture notes, I used m for the number of observations in order to differentiate from n which is reserved for the dimensionality of the argument of the objective function

$$f(x), \ x \in \mathbb{R}^n$$

In this slide deck, I use n for the number of observations, in order for the calculations to align with standard calculations on computational in the literature

The notation $\mathcal{O}(\cdot)$ used in subsequent slides indicates the order of computations required

Computational Steps for Least Squares

Computing the least squares solution requires:

- **Step 1:** Compute $X^\top X \in \mathbb{R}^{p\times p}$
  - To obtain each entry in this $p \times p$ entry we need to multiply two n-dimensional vectors;
    the vector that contains the n-observations for variable i with the vector that contains the n-observations for variable j, with $i, j = 1, \cdots, p$
  - Each such vector multiplication requires $\mathcal{O}(n)$ operations
  - Total: $\mathcal{O}(np^2)$ operations,
    since $X^\top X$ has p^2 entries and each entry requires $\mathcal{O}(n)$ operations
- **Step 2:** Compute $X^\top y \in \mathbb{R}^p$
  - To obtain each $(X^\top y)$ we need to multiply again two n-dimensional vectors; the vector that contains the $n$observations for each variable, with the n-dimensional response vector y
  - Total: $\mathcal{O}(np)$ operations

Computational Steps for Least Squares (Continued)

- **Step 3:** Solve the system $X^\top X\beta = X^\top y$
  - Typically requires computing the inverse $(X^\top X)^{-1}$
  - Matrix inversion requires $\mathcal{O}(p^3)$ operations
  - In practice, we typically solve using Cholesky decomposition, QR decomposition, or eigenvalue decomposition for better numerical stability
  - These methods also require $\mathcal{O}(p^3)$ operations
- **Total computational complexity:**

$$\boxed{\mathcal{O}(np^2 + p^3)} \tag{3}$$

Computational Burden for Large Data sets

Consider a large-scale problem with:

$$n = 10^6 \text{ (samples)}, \quad p = 10^5 \text{ (features)} \tag{4}$$

- $np^2 = 10^6 \cdot (10^5)^2 = 10^{16}$ operations
- $p^3 = (10^5)^3 = 10^{15}$ operations

The dominant term is $\mathcal{O}(np^2) = \boxed{10^{16}}$ operations

- Modern CPUs can perform $\sim 10^9$ operations per second
- This would take approximately 10^7 seconds ≈ 115 days!
- Memory requirements would be enormous as well

Gradient Descent for Least Squares

Instead of computing the closed-form solution, we can iteratively approach the minimum:

- Initialize β_0 (often as zeros or small random values)
- Update rule:

$$\beta_{k+1} = \beta^{(t)} - \eta \nabla_\beta \mathrm{SSE}(\beta) \tag{5}$$

  where $\eta > 0$ is the step size
- The gradient of the SSE function is given by:

$$\nabla_\beta \frac{1}{2n} \|X\beta - y\|_2^2 = \frac{1}{n} \left[X^\top (X\beta - y) \right] \tag{6}$$

Computational Steps per Iteration

Each iteration of gradient descent requires:

- **Step 1:** Compute $X\beta \in \mathbb{R}^n$
  - Each entry multiplies two p-dimensional vectors and there are n such entries
  - Total: $\mathcal{O}(np)$ operations
- **Step 2:** Compute residual $r = X\beta - y \in \mathbb{R}^n$
  - Simple vector subtraction
  - Total: $\mathcal{O}(n)$ operations

Computational Steps per Iteration (Continued)

- **Step 3:** Compute $X^\top r \in \mathbb{R}^p$
  - Each entry requires multiplication of two n-dimensional vectors and there are p such entries
  - Total: $\mathcal{O}(np)$ operations
- **Step 4:** Update β
  - Simple vector subtraction
  - Total: $\mathcal{O}(p)$ operations
- **Total computational complexity per iteration:**

$$\boxed{\mathcal{O}(np)} \tag{7}$$

Computational Advantage for Large Data sets

For our large-scale problem with:

$$n = 10^6 \text{ (samples)}, \quad p = 10^5 \text{ (features)} \tag{8}$$

- Cost per iteration: $np = 10^6 \cdot 10^5 = \boxed{10^{11}}$ operations
- With a good choice of step size, convergence typically requires $10^2 - 10^3$ iterations
- Total cost: $\sim 10^{13} - 10^{14}$ operations

Gradient descent offers a computational advantage of $\sim 10^2 - 10^3$ times over the closed-form solution for large-scale problems

Conditions for Convergence

For gradient descent to find the optimal regression coefficients:

- The matrix $X^\top X$ must be **positive definite**
  - Guarantees the existence of a unique global minimum
  - This is the same condition needed for $(X^\top X)^{-1}$ to exist in the closed-form solution
- The Hessian of the SSE function is $\frac{1}{n}(X^\top X)$
- If $X^\top X$ is positive definite, gradient descent with an appropriate step size will converge to the global minimum

Step Size Selection

- Theoretical bound: For convergence, the step size η must satisfy:

$$0 < \eta < \frac{2}{\lambda_{\max}(X^\top X)} \tag{9}$$

  where $\lambda_{\max}(X^\top X)$ is the largest eigenvalue of $X^\top X$
- Computing $\lambda_{\max}(X^\top X)$ requires $\mathcal{O}(p^3)$ operations
  - This defeats the purpose of using gradient descent to avoid $\mathcal{O}(p^3)$ computations
- Practical solution: **Backtracking line search**
  - Start with a relatively large step size
  - Decrease it gradually until a sufficient decrease condition is met
  - No need to compute eigenvalues explicitly

Computational Cost Comparison

Method	**Computational Complexity**
Closed-form Least Squares	$\mathcal{O}(np^2 + p^3)$
Gradient Descent (per iteration)	$\mathcal{O}(np)$
Gradient Descent (total, k iterations)	$\mathcal{O}(k \cdot np)$

For $n = 10^6$ and $p = 10^5$:

- Closed-form: $\sim 10^{16}$ operations
- Gradient Descent (1000 iterations): $\sim 10^{14}$ operations

For large-scale problems, gradient descent offers a significant computational advantage.

When to Use Each Method

Use closed-form solution when:

- p is small (few features)
- Exact solution is required
- Numerical stability isn't a significant concern
- Available computational resources can handle $\mathcal{O}(np^2 + p^3)$

Use gradient descent when:

- p is large (many features)
- Approximate solution is acceptable
- The problem is "well-conditioned" for faster convergence
- Memory constraints prevent storing or operating on $X^\top X$

Other Considerations

Beyond basic gradient descent:

- **Momentum methods:** Polyak, Nesterov
- **Adaptive learning rates:** AdaGrad, ADAM, ADAM-W
- **Stochastic variants:** Stochastic gradient
  - Further computational savings with subsampling
  - Ideal for extremely large datasets ($n \gg p$)
- **Coordinate descent:** Update one coordinate at a time (will be covered after Exam I)
  - Particularly efficient when X is sparse

Numerical stability:

- Closed-form solution may suffer from ill-conditioning
- Gradient methods can be more robust with appropriate step sizes