EE 270: Large Scale Matrix Computation and Optimization **Winter 2020**

Lecture 14 — February 20

Lecturer: Mert Pilanci *Scribe: Elaina Chai*

In this lecture, the following points will be covered:

- Summary of methods to optimize the Least Sqaures Cost. We will consider the trade offs of these methods in terms of their computational complexity.
- Applying Random Projection to Newton's Method
- Generalizing Gradient Descent to Strongly Convex Functions

14.1 Introduction: Optimizing the Least Squares Cost

For $A \in \mathbf{R}^{n \times d}$, consider optimizing the least squares cost:

$$\min_x f(x) = \min_x \frac{1}{2}\|Ax - b\|_2^2 \tag{14.1}$$

Equation 14.1 has gradient $\nabla f(x) = A^T(Ax - b)$ and Hessian $\nabla^2 f(x) = A^T A$. κ is the condition number of $A^T A$.

In Lecture 13, three methods were introduced to optimize 14.1, with fixed step size μ_t:

- Gradient Descent (GD)
  - $x_{t+1} = x_t - \mu_t \nabla f(x_t) = x_t - \mu_t A^T(Ax_t - b)$
- Gradient Descent with Momentum (GDM), a.k.a. the Heavy Ball Method [1]
  - $x_{t+1} = x_t - \mu_t \nabla f(x_t) + \beta_t(x_t - x_{t-1}) = x_t - \mu_t A^T(Ax_t - b) + \beta_t(x_t - x_{t-1})$
- Newton's Method
  - $x_{t+1} = x_t - \mu_t(\nabla^2 f(x_t))^{-1}\nabla f(x_t) = x_t - \mu_t(A^T A)^{-1}A^T(Ax_t - b)$

[1] Typical GD has a fixed step size set by μ_t. In GDM, the term $\beta_t(x_t - x_{t-1})$ is known as momentum. While momentum helps to improve convergence speed, momentum can lead to overshooting the minimum point, as well as instability. Errors due to the stochastic nature of $\nabla f(x)$, as well as from sources of noise such as quantization, are at much greater risk of amplification from momentum. Therefore, convergence proofs with momentum should be treated more like guidelines.

14.2 Computational Complexity and Trade-offs

For ϵ accuracy in the objective value, i.e. $\|A\hat{x} - Ax^*\|_2 \leq \epsilon$:

- Gradient Descent (GD)
  - Computational Cost of $\kappa nd \log(\frac{1}{\epsilon})$ for ϵ accuracy.
- Gradient Descent with Momentum (GDM), a.k.a. the Heavy Ball Method
  - Computational Cost of $\sqrt{\kappa} nd \log(\frac{1}{\epsilon})$ for ϵ accuracy.
  - The eigenvalues of $A^T A$ is needed to determine optimal step size.
- Conjugate Gradient Descent: A variant of GD in which the step direction is orthogonal to the previous step. This is not covered in detail EE270
  - Computational Cost of $\sqrt{\kappa} nd \log(\frac{1}{\epsilon})$ for ϵ accuracy.
- Newton's Method
  - Computational Cost of $O(nd^2) + O(d^3)$. The first term is to form the Hessian, and the second term is the convert to invert the Hessian.

14.3 Newton's Method with Random Projection

14.3.1 Newton's Method vs Gradient Descent: A Taylor Approximation Viewpoint

Consider a generic twice differentiable function f. Apply a second order Taylor approximation at a point x_t:

$$f(y) \approx f(x_t) + \nabla f(x_t)^T (y - x_t) + \frac{1}{2}(y - x_t)\nabla^2 f(x_t)(y - x_t) \tag{14.2}$$

Minimizing (14.2) gives Newton's Method:

$$x_{t+1} = x_t - \mu_t \big(\nabla^2 f(x)\big)^{-1} \nabla f(x) \tag{14.3}$$

$\big(\nabla^2 f(x)\big)^{-1} \nabla f(x)$ is the direction of the step, and $\big(\nabla^2 f(x)\big)^{-1}$ is the iverse Hessian, with dimensions $d \times d$. Set f to a loss function such as the least squares cost function: $f(Ax) = \|Ax - b\|_2$, where $A \in \mathbf{R}^{n \times d}$. Forming the Hessian takes complexity $O(nd^2)$, and inverting it takes $O(d^3)$.[2]

By comparison, GD is far less complex to compute. We can view GD as the Taylor Approximation with the Identity Matrix as the Hessian (This is equivalent to removing the curvature information from (14.3)). By using the Identity matrix instead, GD automatically saves $O(nd^2) + O(d^3)$.

[2]This analysis will also hold for the Logistic Regression, in which the Squared Loss is replaced with the Cross Entropy Loss.

14.3.2 Applying the Random Projection

An important advantage of Newton over GD, is that Newton can converge in one step. However a massive roadblock is the increased complexity to calculate the inverted Hessian. This makes Newton's Method a possible application for Sketching methods. We will apply Random Projections to reduce the computational complexity of calculating $\left(A^TA\right)^{-1}$. For a fixed step size μ, and a sketch matrix $S \in \mathbf{R}^{m\times n}$, we get Randomized Newton's Method:

$$x_{t+1} = x_t - \mu(A^TS^TSA)^{-1}A^T(Ax-b) \tag{14.4}$$

The type of sketch will be determined by the number of samples we want.

Computational Cost

Use Fast Johnson Lindenstrauss Transform (FJLT) to form SA, with complexity $O(nd\log n)$.

To compute $(A^TS^TSA)^{-1}$, we can directly invert (A^TS^TSA) at a cost of $O(d^3)$. The only issue is if (A^TS^TSA) is not invertible. A trick to get around this is to instead invert $(A^TS^TSA+\alpha I)$ (this is always invertible).

Alternatively, to compute $(A^TS^TSA)^{-1}$, we can factorize SA using QR or SVD. To use SVD, consider $SA = U\Sigma V^T$. Then $(A^TS^TSA)^{-1} = (V\Sigma^2V^T)^{-1} = V\Sigma^{-2}V^T$. Factorizing SA, at a cost of $O(md^2)$, can be more efficient for very small m.

A Regularization Trick to Ensure Invertibility

A trick to computing $(A^TS^TSA)^{-1}$ if (A^TS^TSA) is not invertible, is to instead form $(A^TS^TSA+\alpha I)^{-1}$ (which is always invertible) at a cost of $O(md^2)$:

$$(A^TS^TSA+\alpha I)^{-1} = (V(\Sigma^2+\alpha I)V^T)^{-1} = V(\Sigma^2+\alpha I)^{-1}V^T \tag{14.5}$$

The trick comes from regularization: $\min_x \|Ax-b\|_2^2 + \lambda\|x\|_2^2$. The Hessian is $\nabla^2 f(x) = 2A^TA + 2\lambda I \approx 2A^TS^TSA + 2\lambda I$.

14.3.3 Analyzing Newton's Method with Random Projection

In this section, we will analyze the eigenvalues, convergence rate and computational complexity of (14.4). Define:

$$\Delta_t = A(x_t - x^*) \tag{14.6}$$

where x^* is the solution. If we apply this equation to (14.4), then:

$$\Delta_{t+1} = \Delta_t - \mu A(A^TS^TSA)^{-1}A^T\Delta_t \tag{14.7}$$

After M iterations:

$$\Delta_M = (I - \mu A(A^T S^T S A)^{-1} A^T)^M \Delta_0 \tag{14.8}$$

We want the above system to converge as quickly as possible. Let $A = U\Sigma V^T$. Equation (14.6) implies $\Delta_t \in Range(A)$, therefore $UU^T\Delta_t = \Delta_t$ and $\|U^T\Delta_t\|_2 = \|\Delta_t\|_2$. Applying all of this to (14.8):

$$\Delta_M = (I - \mu U(U^T S^T S U)^{-1} U^T)^M \Delta_0 \tag{14.9}$$

$$\begin{aligned} U^T\Delta_M &= U^T(I - \mu U(U^T S^T S U)^{-1} U^T)^M U U^T \Delta_0 \\ &= (I - \mu(U^T S^T S U)^{-1})^M U^T \Delta_0 \\ &\leq (I - \mu(U^T S^T S U)^{-1})^M \|U^T\Delta_0\|_2 \end{aligned} \tag{14.10}$$

$$\|\Delta_M\|_2 \leq \sigma_{\max}((I - \mu(U^T S^T S U)^{-1})^M)\|\Delta_0\|_2 \tag{14.11}$$

$$= \max_{i=1,\ldots,d} |I - \mu\lambda_i(U^T S^T S U)^{-1})|^M \|\Delta_0\|_2 \tag{14.12}$$

Eigenvalues and the Optimal Step Size

For $U^T U = I$, we can approximate this matrix multiplication with a random projection matrix and define an error bound: $\|U^T U - U^T S^T S U\|_F \leq \epsilon$. This implies:

$$\sigma_{\max}(I - U^T S^T S U) \leq \epsilon \tag{14.13}$$

Equation (14.13) is identical to $1 - \lambda_i(U^T S^T S U) \leq \epsilon \ \forall i = 1, ..., d$, suggesting that all eigenvalues of $U^T S^T S U$ are in the range $[1-\epsilon, 1+\epsilon]$ and hence all eigenvalues of $(U^T S^T S U)^{-1}$ are in the range $[\frac{1}{1+\epsilon}, \frac{1}{1-\epsilon}]$. If we plug the extremes of this range back into (14.11), then:

$$\|\Delta_M\|_2 \leq \max\left(\left|1 - \mu\frac{1}{1-\epsilon}\right| \cdot \left|1 - \mu\frac{1}{1+\epsilon}\right|\right)^M \|\Delta_0\|_2 \tag{14.14}$$

The optimal step size that minimizes this upper bound satisfies:

$$\left|1 - \mu^*\frac{1}{1-\epsilon}\right| = \left|1 - \mu^*\frac{1}{1+\epsilon}\right| \tag{14.15}$$

where $\mu^* = \frac{2}{\frac{1}{1-\epsilon} + \frac{1}{1+\epsilon}} = (1-\epsilon)(1+\epsilon)$. Notice that as ϵ decreases, the μ^* converges to the optimal μ for Newton's Method.

Convergence Rate

If we plug μ^* back into (14.14):

$$\begin{aligned}\|\Delta_M\|_2 &\leq \max\left(\left|1-\mu\frac{1}{1-\epsilon}\right|\cdot\left|1-\mu\frac{1}{1+\epsilon}\right|\right)^M\|\Delta_0\|_2\\ &= \max(|1-(1+\epsilon)|,|1-(1-\epsilon)|)^M\|\Delta_0\|_2\\ &= \epsilon^M\|\Delta_0\|_2 \end{aligned} \tag{14.16}$$

Computational Complexity

To avoid overloading, let us change notation in (14.16) to replace $\epsilon \to \alpha$.

It takes $O(nd\log n)$ to form SA using FJLT. Forming A^TS^TSA and $(A^TS^TSA)^{-1}$takes $O(md^2)$ and $O(d^3)$ respectively. Running M iterations of $A^T(Ax-b)$ takes $O(ndM)$. Let $\alpha = 0.1 \to (\alpha)^M = 10^{-M}$. If we want $10^{-M} < \epsilon$, where ϵ is in this case the error bound, then $M = \log\frac{1}{\epsilon}$. The total computational complexity is $nd\log n + d^3 + ndM = nd\log n + d^3 + nd\log(\frac{1}{\epsilon})$.

Compared to other methods we have seen before (with $\|A\hat{x} - Ax^*\|_2 \leq \epsilon$), there is no clear better method. The choice of method will depend on the operating regime. If we want to avoid iterations altogether, Left Sketch is better, but it always has comparatively low accuracy.

Large n makes ϵ small, and therefore Random Newton's Method (RNM) is better. RNM has very high accuracy, as we can change ϵ by changing the number of iterations. Furthermore, we have lost our dependence on the condition number κ. At large d, κ is small, and Conjugate Gradient Descent (CGD) is better. However Both RNM and CGD require repeated views of the full A.

14.4 Generalizing Gradient Descent to Strongly Convex Problems

We can expand Gradient Descent with and without momentum to strongly convex functions.

A convex function is strongly convex if there exists two positive constants, $\beta_- \leq \beta_+$ such that, for every x in the domain of f:

$$\beta_- \leq \lambda_i(\nabla^2 f(x)) \leq \beta_+$$

This is equivalent to limiting the stretch on the contours of f. $\lambda_{\min}(\nabla^2 f(x)) \geq \beta_-$ is a lower bound on the curvature, and $\lambda_{\max}(\nabla^2 f(x)) \leq \beta_+$ is an upper bound on the curvature.

14.4.1 Gradient Descent

Define $x_{t+1} = x_t - \mu_t\nabla f(x_t)$, and suppose f is strongly convex with parameters β_-, β_+. Let $f^* := \min_x f(x)$, and set a constant step size of $\mu_t = \frac{1}{\beta_+}$:

$$f(x_{t+1}) - f^* \leq (1 - \frac{\beta_-}{\beta_+})(f(x_t) - f^*) \tag{14.17}$$

Applied recursively for M steps, we get:

$$f(x_M) - f^* \leq (1 - \frac{\beta_-}{\beta_+})^M(f(x_0) - f^*) \tag{14.18}$$

$(f(x_t) - f^*)$ gives the initial gap between f and the minimum point. This gap contracts according to the bounds on the Hessian set by parameters β_-, β_+. For a perfect quadratic, $\beta_- = \beta_+$ and the convergence takes one step.

For optimizing functions $f(Ax)$, where $A \in \mathbf{R}^{n\times d}$, the computation complexity is $O(\kappa nd \log(\frac{1}{\epsilon}))$ and $\kappa = \frac{\beta_+}{\beta_-}$ is the generalized condition number.

14.4.2 Gradient Descent with Momentum

Let us apply momentum. The analysis for strongly convex functions parallels Least Squares. In this case, the update on x_{t+1} picks up the momentum term:

$$x_{t+1} = x_t - \mu\nabla f(x_t) + \beta(x_t - x_{t-1}) \tag{14.19}$$

The step size is $\mu = \frac{4}{(\sqrt{\beta_+}+\sqrt{\beta_-})^2}$. The momentum parameter is $\beta = \max(|1-\sqrt{\mu\beta_-}|, |1-\sqrt{\mu\beta_+}|)^2$. For optimizing functions $f(Ax)$, where $A \in \mathbf{R}^{n\times d}$, the computation complexity is $O(\sqrt{\kappa}nd \log(\frac{1}{\epsilon}))$ and $\kappa = \frac{\beta_+}{\beta_-}$. Notice that just like the least squares analysis, the dependence on κ has reduced compared to the gradient descent version.