CS189 (SPRING 2023) NOTES

A PREPRINT

Chawin Sitawarin *

University of California, Berkeley

1 Discussion 3

1.1 Solving Linear Regression with Gradient Descent

"Why do we care so much about eigenvalues?"

There are two billion reasons, but today we will look at one of them.

Let's consider a quadratic function of the form

$$f(w) = \frac{1}{2} \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|^2 \tag{1}$$

$$f(w) = \frac{1}{2} \boldsymbol{w}^\top \boldsymbol{Q} \boldsymbol{w}^\top + \boldsymbol{w}^\top \boldsymbol{a} + b \tag{2}$$

Note that in this special setting we have $\boldsymbol{Q} = \boldsymbol{X}^\top \boldsymbol{X}$ and $\boldsymbol{a} = -\boldsymbol{X}^\top \boldsymbol{y}$. Importantly, $\boldsymbol{Q}$ is a positive semidefinite matrix, i.e., $\forall \boldsymbol{X}, \boldsymbol{Q} \succeq 0$. Let's also further assume that $\boldsymbol{X}^\top \boldsymbol{X}$ is full rank so we have a stronger condition that $\boldsymbol{Q}$ is positive definite, i.e., $\boldsymbol{Q} \succ 0$.

Linear regression wants to solve

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \frac{1}{2} \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|^2 \tag{3}$$

$$= \arg\min_{\boldsymbol{w}} \frac{1}{2} \boldsymbol{w}^\top \boldsymbol{Q} \boldsymbol{w} + \boldsymbol{w}^\top \boldsymbol{a} + b \tag{4}$$

We already know of this solution (analytically):

$$\nabla_{\boldsymbol{m}} f(\boldsymbol{w}^*) = \boldsymbol{Q}\boldsymbol{w}^* + \boldsymbol{a} = 0 \tag{5}$$

$$\boldsymbol{w}^* = -\boldsymbol{Q}^{-1}\boldsymbol{a} \tag{6}$$

$$= (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y} \tag{7}$$

However, inverting an $n \times n$ matrix is $\mathcal{O}(n^3)$, which is rather expensive for a large n, i.e., high-dimensional data. Note that realistically, while Guassian Elimination involves $\mathcal{O}(n^3)$ operations, it does not sufficiently account for the number of bits needed to represent these intermediate values.

1. Finding matrix inverse with Gaussian Elimination: https://en.wikipedia.org/wiki/Gaussian_elimination#Finding_the_inverse_of_a_matrix
2. Why it is actually $\mathcal{O}(n^5)$ instead of $\mathcal{O}(n^3)$): https://en.wikipedia.org/wiki/Gaussian_elimination#Computational_complexity and https://rjlipton.wpcomstaging.com/2015/01/14/forgetting-results/

*Corresponding email: chawins@berkeley.edu

In short, we rarely want to invert a matrix in practice.

Now we will consider the gradient descent algorithm for solving linear regression. We will denote the eigenvalues of $\boldsymbol{Q}$ as $\lambda_1 > \lambda_2 > \ldots > \lambda_n > 0$ (since we assume that $\boldsymbol{Q}$ is full rank).

$$f(w_{t+1}) = f(w_t - \alpha \nabla f(w_t)) \tag{8}$$

$$= f(w_t) - \alpha \nabla f(w_t)^\top \nabla f(w_t) + \frac{\alpha^2}{2} \nabla f(w_t)^\top \nabla^2 f(\zeta_{w_t,\alpha}) \nabla f(w_t) \qquad \text{(MVT)} \tag{9}$$

$$= f(w_t) - \alpha \left\|\nabla f(w_t)\right\|^2 + \frac{\alpha^2}{2} \nabla f(w_t)^\top Q \nabla f(w_t) \tag{10}$$

$$\leq f(w_t) - \alpha \left\|\nabla f(w_t)\right\|^2 + \frac{\alpha^2 \lambda_1}{2} \left\|\nabla f(w_t)\right\|^2 \qquad \text{(Max. eigenvalue of } \boldsymbol{Q}\text{)} \tag{11}$$

$$\tag{12}$$

Choosing an optimal step size $\alpha = 1/\lambda$, we get

$$f(w_{t+1}) \leq f(w_t) - \frac{\alpha}{2} \left\|\nabla f(w_t)\right\|^2 \tag{13}$$

Note that choosing $\alpha > 1/\lambda$ will mean that the gradient descent algorithm will diverge. So knowing the largest eigenvalue is already important for choosing a good step size and hence ensuring that the gradient descent algorithm will converge.

We can end our proof here and get a convergence rate of $\mathcal{O}(1/T)$, but we can do better under the assumption that $\boldsymbol{Q}$ is PD.

$$f(w_{t+1}) \leq f(w_t) - \frac{\alpha}{2} \left\|\nabla f(w_t)\right\|^2 \tag{14}$$

$$f(w_{t+1}) \leq f(w_t) - \alpha \lambda_n \left(f(w_t) - f^*\right) \qquad \text{(Polyak-Lojasiewicz condition)} \tag{15}$$

$$f(w_{t+1}) - f^* \leq f(w_t) - f^* - \alpha \lambda_n \left(f(w_t) - f^*\right) \tag{16}$$

$$f(w_{t+1}) - f^* \leq \left(1 - \alpha \lambda_n\right) \left(f(w_t) - f^*\right) \tag{17}$$

$$f(w_T) - f^* \leq \left(1 - \alpha \lambda_n\right)^T \left(f(w_0) - f^*\right) \tag{18}$$

$$f(w_T) - f^* \leq \exp\left(-\alpha \lambda_n T\right) \left(f(w_0) - f^*\right) \qquad (1 + x \leq e^x) \tag{19}$$

$$= \exp\left(-\frac{\lambda_n}{\lambda_1} T\right) \left(f(w_0) - f^*\right) \tag{20}$$

This means that the gradient descent algorithm converges exponentially fast at a rate of $\mathcal{O}(\exp(-T))$. And finally, the closer λ_n is to λ_1, the faster the gradient descent algorithm will converge.

In general, the *condition number* of the problem is defined as:

$$\kappa = \frac{\lambda_1}{\lambda_n} \tag{21}$$

"Poorly conditioned" problems have a large condition number, and "well-conditioned" problems have a small condition number (close to 1). Even when the rate is exponential but the condition number is very large, the gradient descent algorithm can still converge very slowly (in the worst case).

In general, if we are dealing with other non-quadratic function, we can still replace $\boldsymbol{Q}$ with the Hessian matrix $\boldsymbol{H}(\boldsymbol{w}) \coloneqq \nabla^2 f(\boldsymbol{w})$, and everything we talk about still holds. λ_1 corresponds to the L-Lipschitz constant of the gradients of f. This generally tells how smooth f is, and larger L means that f is less smooth (gradients change fast in some directions). On the other hand, λ_n corresponds to the "strong" convexity of f, a stronger condition than convexity.

Reference: https://www.cs.cornell.edu/courses/cs4787/2020sp/lectures/Lecture2.pdf