EE270: Large Scale Matrix Computation and Optimization
February 18th, 2020
Lecture 13: Gradient Descent with Momentum and Preconditioning
Scribe: Chandra Rajyam

13.1 - Optimizing Convex Least Squares Cost

Recall the least squares convex optimization problem:

$$min_x \frac{1}{2} \| Ax - b \|_2^2$$

The gradient is:

$$\nabla f(x) = A^T(Ax - b)$$

The gradient descent update rule is:

$$x_{t+1} = x_t - \mu A^T(Ax_t - b)$$

In this case, the step size is fixed:

$$\mu_t = \mu$$

The optimal value (which minimizes the objective function) is x*. Hence, by convexity:

$$\nabla f(x^*) = A^T(Ax^* - b) = 0$$

The error at each time step is defined as:

$$\Delta_t = x_t - x^*$$

So, it follows that:

$$\Delta_{t+1} = \Delta_t - \mu A^T(Ax_t - b) = \Delta_t - \mu A^T(Ax_t - b) + \mu A^T(Ax^* - b) = \Delta_t - \mu A^T A \Delta_t$$

Should we run the gradient descent algorithm for M iterations:

$$\Delta_M = (I - \mu A^T A)^M \Delta_0$$

Should we take the Euclidean norm of both sides, we can take advantage of the fact that the operator norm coincides with the largest singular value:

$$\| \Delta_M \|_2 \leq \sigma_{max}((I - \mu A^T A)^M) \| \Delta_0 \|_2$$

Since we are dealing with a symmetric matrix:

$$\sigma_{max}((I - \mu A^T A)^M) = max_{i=1,..,d} |1 - \mu \lambda_i(A^T A)|^M$$

Where λ_i is the i-th eigenvalue in decreasing order.

We now make the following definitions:

λ_- is the smallest eigenvalue of $A^T A$

λ_+ is the largest eigenvalue of $A^T A$

Now, it follows that:

$$max_{i=1,..,d}|1-\mu\lambda_i(A^TA)| = \max(|1-\mu\lambda_-|, |1-\mu\lambda_+|)$$

The optimal step size would be chosen to minimize the above (in order to best minimize the error). In other words:

$$\mu_{opt} = min_{\mu\geq 0}\max(|1-\mu\lambda_-|, |1-\mu\lambda_+|)$$

It also worthwhile to note that both values in the pair would have to be equal when μ is optimal. Hence:

$$\mu_{opt} = \frac{2}{\lambda_+ + \lambda_-}$$

Now we can go about finding the convergence rate of the gradient descent algorithm:

$$\max(|1-\mu\lambda_-|, |1-\mu\lambda_+|) = \frac{\lambda_+ - \lambda_-}{\lambda_+ + \lambda_-}$$

Now (after we substitute the above):

$$\| \Delta_M \|_2 \leq \left(\frac{\lambda_+ - \lambda_-}{\lambda_+ + \lambda_-}\right)^M \| \Delta_0 \|_2$$

Depending on the eigenvalues of A^TA, we have different convergence rate. If A^TA has identical eigenvalues, we have one-step convergence. However, should the largest eigenvalue >> smallest eigenvalue, we have slow convergence.

We now define the condition number:

$$\kappa := \frac{\lambda_+}{\lambda_-}$$

Hence:

$$\| \Delta_M \|_2 \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^M \| \Delta_0 \|_2$$

Say we initialize at $x_0 = 0$, and we desire an accuracy $\| \Delta_M \|_2 \leq \epsilon$. We would then need our number of iterations M to be:

$$Mlog\left(\frac{\kappa - 1}{\kappa + 1}\right) + log \| x^* \|_2 \leq \log(\epsilon)$$

$$M = O\left(\frac{\log(\frac{1}{\epsilon})}{\log(\frac{\kappa + 1}{\kappa - 1})}\right)$$

It is worthwhile to note that $\log\left(\frac{\kappa+1}{\kappa-1}\right) \approx \frac{2}{\kappa-1}$ for large κ. So (for large κ) the computation cost is:

$$M = O(\kappa\log(\frac{1}{\epsilon}))$$

13.2 – Momentum

We now modify the gradient descent update rule to:

$$x_{t+1} = x_t - \mu_t \nabla f(x_t) + \beta_t(x_t - x_{t-1})$$

The additional term is known as momentum. It is related to the discretization of the second order ODE (modelling the motion of a body in a potential field given by f):

$$\ddot{x} + a\dot{x} + b\nabla f(x)$$

Momentum is also called accelerated gradient descent, or the heavy-ball method. It can be rewritten as:

$$p_t = \beta_t p_{t-1} - \nabla f(x_t)$$
$$x_{t+1} = x_t + \alpha_t p_t$$

In this case, p_t is the search direction. This update rule has short term memory. We also typically set $p_0 = 0$.

13.2.1 – Momentum for Least Squares

Recall from 13.1 that:

$$\Delta_{t+1} = \Delta_t - \mu A^T A \Delta_t$$

Since there is one-time step memory, consider:

$$V_t := \| \Delta_{t+1} \|_2^2 + \| \Delta_t \|_2^2$$

It is worthwhile to note that V_t upper bounds error (Lyapunov Analysis):

$$\| \Delta_t \|_2^2 \leq V_t$$

13.2.2 – Convergence Analysis

Recall the least squares problem, update rule and error definition once more:

$$min_x \frac{1}{2} \| Ax - b \|_2^2$$
$$x_{t+1} = x_t - \mu_t \nabla f(x_t) + \beta_t (x_t - x_{t-1})$$
$$\Delta_t = x_t - x^* \text{ where } x^* = A^\dagger b$$

Notice the following:

$$b = Ax^* + b^\perp$$
$$\nabla f(x_t) = A^T A \Delta_t$$
$$\text{since } \nabla f(x^*) = A^T (Ax^* - b) = 0$$

We can use those equations to write:

$$\begin{bmatrix} \Delta_{t+1} \\ \Delta_t \end{bmatrix} = \begin{bmatrix} x_t - \mu_t \nabla f(x_t) + \beta_t (x_t - x_{t-1}) - x^* \\ \Delta_t \end{bmatrix}$$
$$= \begin{bmatrix} (1+\beta)I - \alpha A^T A & \beta I \\ I & 0 \end{bmatrix} [\begin{matrix} \Delta_t \\ \Delta_{t+1} \end{matrix}]$$

We now iterate for i = 1,…,M:

$$\begin{bmatrix} \Delta_{M+1} \\ \Delta_M \end{bmatrix} = \begin{bmatrix} (1+\beta)I - \alpha A^T A & \beta I \\ I & 0 \end{bmatrix}^M [\begin{matrix} \Delta_M \\ \Delta_{M+1} \end{matrix}]$$

Similar to before, we take l2 norms:

$$\| \begin{bmatrix} \Delta_{M+1} \\ \Delta_M \end{bmatrix} \| \leq \sigma_{max}(\begin{bmatrix} (1+\beta)I - \alpha A^T A & \beta I \\ I & 0 \end{bmatrix}^M) \| [\begin{matrix} \Delta_M \\ \Delta_{M+1} \end{matrix}] \|$$

13.2.2.1 – Spectral Radius

Say there is a d by d matrix with eigenvalues $\lambda_1, \ldots, \lambda_d$. Then, the spectral radius is defined as:

$$\rho(M) := max_{i=1,\ldots,d} |\lambda_i|$$

Lemma:

$$lim_{k\to\infty}\sigma_{max}(M^K)^{1/k} = \rho(M)$$

13.2.2 – Convergence Analysis

Now we return to convergence analysis. Let λ_i denote the eigenvalues of A^TA for i=1,..,d.

Lemma:

The eigenvalues of:

$$\begin{bmatrix}(1+\beta)I-\alpha A^TA & \beta I\\ I & 0\end{bmatrix}$$

are given by the eigenvalues of 2 x 2 matrices:

$$\begin{bmatrix}(1+\beta)-\alpha\lambda_i & -\beta\\ 1 & 0\end{bmatrix}$$

for i=1,..,d.

We set α and β to minimize the spectral radius:

$$\alpha = \frac{4}{\sqrt{\lambda_+} + \sqrt{\lambda_-}}$$

$$\beta = \frac{\sqrt{\lambda_+} - \sqrt{\lambda_-}}{\sqrt{\lambda_+} + \sqrt{\lambda_-}}$$

The spectral radius then becomes:

$$\rho\left(\begin{bmatrix}(1+\beta)I-\alpha A^TA & \beta I\\ I & 0\end{bmatrix}\right) = \frac{\sqrt{\lambda_+} - \sqrt{\lambda_-}}{\sqrt{\lambda_+} + \sqrt{\lambda_-}}$$

The convergence result becomes:

$$\| \begin{bmatrix}\Delta_{M+1}\\ \Delta_M\end{bmatrix} \| \le \left(\frac{\sqrt{\lambda_+} - \sqrt{\lambda_-}}{\sqrt{\lambda_+} + \sqrt{\lambda_-}}\right)^M \| \begin{bmatrix}\Delta_M\\ \Delta_{M+1}\end{bmatrix} \|$$

This is better than regular gradient descent as the complexity is:

$$\sqrt{\kappa}nd \log\left(\frac{1}{\varepsilon}\right)$$

As opposed to (when β=0):

$$\kappa nd \log\left(\frac{1}{\varepsilon}\right)$$

13.3 – Newton's Method

Recall the second order Taylor's approximation:

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

Say we want to minimize the approximation. The update rule is:

$$x_{t+1} = x_t - \mu_t\left(\nabla^2 f(x_t)\right)^{-1}\nabla f(x_t)$$

The complexity (for minimizing f(Ax) where A is n x d) is $O(nd^2)$ to form the Hessian and $O(d^3)$ to invert. Alternatively, $O(nd^2)$ for factorizing the Hessian. It is also worthwhile to note that Newton's method converges for in one step (when the step size is 1).