| | |
|---|---|
| **CS 15-851: Algorithms for Big Data** | Spring 2024 |
| Lecture 4 — 02/08/2024 | |
| *Prof. David Woodruff* | *Scribe: Harry H. Jiang* |

# 1 Low-Rank Approximation

**Reminder:** the goal of LoRA is to output a rank $k$ matrix $A'$ such that $||A-A'||_F \leq (1+\varepsilon)||A-A_k||_F$ where $A_k$ is the $k$-rank approximation of $A$. This can be done in $\texttt{nnz}(A)+(n+d)\texttt{poly}(k/\varepsilon)$ time.

So far, the algorithm is to:

1. Compute $SA$
2. Project each of the rows of $A$ onto $SA$
3. Find best rank-$k$ approximation of projected points inside the rowspace of $SA$.

We know previously that $||A_k(SA_k)^{-}SA - A||_F^2 \leq (1+\varepsilon)||A_k - A||_F^2$. From this, we can determine that

$$\min_{\text{rank-}k \text{ matrix } X} ||XSA - A||_F^2 \leq ||A_k(SA_k)^{-}SA - A||_F^2 \leq (1+\varepsilon)||A - A_k||_F^2 \tag{1}$$

By the normal equations we when have ,

$$||XSA - A||_F^2 = ||XSA - A(SA)^{-}SA||_F^2 + ||A(SA)^{-}SA - A||_F^2 \tag{2}$$

where the last last term is $A$ minus the projection of $A$ onto $SA$ (or, with SVD, $P_{SA} = (SA)^{-}SA = V\Sigma U^T U\Sigma V^T = VV^T$); there exists a difference between it and the left-hand side due to the requirement of $X$ to be of rank $k$. Removing the second term in the above from the minimum, we obtain

$$\min_{\text{rank-}k\ X} ||XSA - A||_F^2 = ||A(SA)^{-}SA - A||_F^2 + \min_{\text{rank-}k\ X} ||XSA - A(SA)^{-}SA||_F^2 \tag{3}$$

We can then apply SVD to $SA$ and write it as $SA = U\Sigma V^T$ as a "thin" SVD (meaning $\Sigma$ is of width rank($SA$) and the bottom rows of $V^T$ are discarded) where $SA \in \mathbb{R}^{x\times d}$, $U \in \mathbb{R}^{s\times s}$, $\Sigma \in \mathbb{R}^{s\times s}$, $V^T \in \mathbb{R}^{s\times d}$ and $s =\texttt{poly}(k/\varepsilon)$ We can notice that

$$\min_{\text{rank-}k\ X} ||XSA - A(SA)^{-}SA||_F^2 = \min_{\text{rank-}k\ X} ||XU\Sigma - A(SA)^{-}U\Sigma||_F^2 \tag{4}$$

where $X \in \mathbb{R}^{n\times s}$ and $U\Sigma \in \mathbb{R}^{s\times s}$. $V^T$ is discarded due to its norm-preserving properties. Performing an $s\times s$ change of basis $Y = XU\Sigma$, we have

$$\min_{\text{rank-}k\ X} ||XSA - A(SA)^{-}SA||_F^2 = \min_{\text{rank-}k\ Y} ||Y - A(SA)^{-}U\Sigma||_F^2 \tag{5}$$

We can then compute the SVD of $A(SA)^{-}U\Sigma$. This is better than naïve SVD, but multiplying $A*(SA)^{-}U\Sigma$ is still slow, thus step 2 in the LoRA algorithm becomes a bottleneck in computation.

### 1.1 Sketching Projection onto $SA$

We use sketching to approximate the projection in order to achieve the desired runtime:

$$\min_{\text{rank-}k\ X} ||XSA - A||_F^2 \to \min_{\text{rank-}k\ X} ||XSAR - AR||_F^2 \tag{6}$$

This can be solved using affine embedding $R$ such that

$$||XSAR - AR||_F^2 = (1 \pm \varepsilon)||XSA - A||_F^2 \forall X \tag{7}$$

where both $AR$ and $SAR$ can be computed in $\texttt{nnz}(A)$ time.

Solving for minimum rank-$k$ matrix $X$, we have:

$$\min_{\text{rank-}k\ X} ||XSAR-AR||_F^2 = ||AR(SAR)^- SAR-AR||_F^2 + \min_{\text{rank-}k\ X} ||XSAR-AR(SAR)^- SAR||_F^2 \tag{8}$$

with change of bases, all we need to compute is

$$\min_{\text{rank-}k\ Y} ||Y - AR(SAR)^- SAR||_F^2 \tag{9}$$

where the minimizer $Y$ should be in the row span of $SAR$ and SVD takes $n\texttt{poly}(k/\varepsilon)$ to compute. Since $Y = XSAR$ for some $X$, we can output $Y(SAR)^- SA$ in factored form which is cheaper to compute:

$$L = U \in \mathbb{R}^{n\times k}, R = \Sigma V^T (SAR)^- SA \in \mathbb{R}^{k\times d} \tag{10}$$

## 2 High Precision Regression

Returning to regression, we aim to find an $x'$ for which $||Ax' - b||_2 \leq (1+\varepsilon)\min_x ||Ax - b||_2$ with high probability. So far, the algorithms covered for regressions run at $\texttt{poly}(d/\varepsilon)$. This may sometimes still be too expensive.

**Goal:** find an algorithm for regression that runs at $\texttt{poly}(d)\texttt{log}(1/\varepsilon)$.

Beyond that, we want to make $A$ "well-conditioned" using a metric $\kappa$

$$\kappa(A) = \frac{\sup\limits_{||x||_2=1} ||Ax||_2}{\inf\limits_{||x||_2=1} ||Ax||_2} \tag{11}$$

where the numerator is the highest singular value of $A$ and the denominator is the lowest. To be "well-conditioned" would then to have a low $\kappa(A)$ as many algorithms' time complexity depend on it; we want to minimize this value to $O(1)$ using sketching.

### 2.1 Small $QR$ Decomposition

Let $S$ be a $(1 + \varepsilon_0)$-subspace embedding for $A$. We compute $SA \in \mathbb{R}^{d^2\times d}$ (the row number is arbitrary as we are dealing with an arbitrary $S$), then an equivalent $QR$-factorization $SA = QR^{-1}$. e$QR$-factorization is to use SVD where $Q = U$ and $R^{-1} = \Sigma V^T$ which allows for $\kappa(SAR) = \kappa(Q) = 1$

**Claim 1.** Given $\kappa(SAR) = \kappa(Q) = 1$,

$$\kappa(AR) = \frac{1+\varepsilon_0}{1-\varepsilon_0} \tag{12}$$

*Proof.* Evaluate singular value bounds of $AR$ and $SAR$.

$\forall x, ||x||_2 = 1, (1-\varepsilon_0)||ARx||_2 \leq ||SARx||_2 = 1$

$\forall x, ||x||_2 = 1, (1+\varepsilon_0)||ARx||_2 \geq ||SARx||_2 = 1$

$$\kappa(AR) = \frac{\sup\limits_{||x||_2=1} ||ARx||_2}{\inf\limits_{||x||_2=1} ||ARx||_2} \leq \frac{1+\varepsilon_0}{1-\varepsilon_0} \tag{13}$$

■

Aside: if we want $\kappa(AR) = 1$, then $AR = U$.

### 2.2 Finding a Constant Factor Solution

Let $S$ be a $(1+\varepsilon_0)$-subspace embedding for $AR$; we can then solve $x_0 = \operatorname{argmin}_x ||SARx - Sb||_2$. If we compute $SA$ first then $SAR$, then the time to compute $R$ and $x_0$ is $\texttt{nnz}(A)+\texttt{poly}(d)$ for constant $\varepsilon_0$, which is really good for gradient descent with an initial point $x_0$.

The gradient descent algorithm is as follows:

$$x_{m+1} \leftarrow x_m + R^T A^T (b - ARx_m) \tag{14}$$

with the second term being the gradient in linear regression.

Looking at the regression task for a step $m$, we see that

$$\begin{aligned} AR(x_{m+1} - x^*) &= AR(x_m + R^T A^T (b - ARx_m) - x^*) && (15)\\ &= ARx_m + ARR^T A^T b - ARR^T A^T ARx_m - ARx^* && (16) \end{aligned}$$

since $x^* = \min_x ||ARx - b||$, we have $R^T A^T b = R^T A^T ARx^*$, therefore

$$\begin{aligned} AR(x_{m+1} - x^*) &= (AR - ARR^T A^T AR)(x_m - x^*) && (17)\\ &= U(\Sigma - \Sigma^3)V^T(x_m - x^*) && (18) \end{aligned}$$

with the last line stemming from an SVD on $AR$: given $AR = U\Sigma V^T$, $AR - ARR^T A^T AR = U\Sigma V^T - U\Sigma V^T V\Sigma U^T U\Sigma V^T = U\Sigma V^T - U\Sigma\Sigma\Sigma V^T$. The singular value matrices at this point have elements

$$\Sigma = \begin{bmatrix} 1+\varepsilon_0 & & \\ & \ddots & \\ & & 1-\varepsilon_0 \end{bmatrix} \quad \Sigma^3 = \begin{bmatrix} \sim 1+3\varepsilon_0 & & \\ & \ddots & \\ & & \sim 1-3\varepsilon_0 \end{bmatrix} \quad \Sigma - \Sigma^3 = \begin{bmatrix} \pm O(\varepsilon_0) & & \\ & \ddots & \\ & & \pm O(\varepsilon_0) \end{bmatrix} \tag{19}$$

Observing the magnitude of a step, we have:

$$\begin{aligned} ||AR(x_{m+1} - x^*)||_2 &= ||(\Sigma - \Sigma^3)V^T(x_m - x^*)||_2 && (20)\\ &= O(\varepsilon_0)||AR(x_m - x^*)||_2 && (21)\\ &= O(\varepsilon_0)^{m+1}||AR(x_0 - x^*)||_2 && (22) \end{aligned}$$

using the property that

$$\begin{aligned} ||(\Sigma - \Sigma^3)V^T(x_m - x^*)||_2 &\leq O(\varepsilon_0)||V^T(x_m - x^*)||_2\\ &\leq \frac{O(\varepsilon_0)}{1-\varepsilon_0}||\Sigma V^T(x_m - x^*)||_2\\ &= O(\varepsilon_0)||U\Sigma V^T(x_m - x^*)||_2\\ &= O(\varepsilon_0)||AR(x_m - x^*)||_2 \end{aligned}$$

If we set $O(\varepsilon_0) = 1/2$ and $m = \log(1\varepsilon_0)$, then $||AR(x_{m+1} - x^*)||_2 = \varepsilon_0||AR(x_0 - x^*)||_2$, and by Pythagorean theorem, between $ARx_0$, $ARx^*$, and $b$, we can show that $||AR(x_{m+1} - x^*)||_2 \leq O(\varepsilon_0)\texttt{OPT}$.

The final tally for runtime is:

1. $\texttt{nnz}(A)+\texttt{poly}(d)$ for finding preconditioner $R$.
2. $\texttt{nnz}(A)+\texttt{poly}(d)$ for $O(1)$ approximation of $x_0$.
3. $O(\log(1/\varepsilon))(\texttt{nnz}(A)+\texttt{poly}(d))$ for gradient descent updates.

Adding them together yields a time complexity of $\texttt{poly}(d)\texttt{log}(1/\varepsilon)$ as desired.