gram schmidt runtime

Andrew Shell

November 2025

Gram Schmidt Orthonormalization Algorithm

The Gram Schmidt Algorithm takes a linearly independent set of vectors $\{v_1, v_2, ..., v_n\} \in V$ and outputs another set of vectors $\{e_1, e_2, ..., e_k\}$ for $k = 1, 2, ..., n$ that is orthonormal.

The $\text{span}\{v_1, v_2, ..., v_n\} = \text{span}\{e_1, e_2, ..., e_n\}$

$$
\begin{aligned}
e_1 &= \frac{v_1}{||v_1||} \\
e_2 &= \frac{v_2 - \langle v_2, e_1\rangle e_1}{||v_2 - \langle v_2, e_1\rangle e_1||} \\
e_3 &= \frac{v_3 - \langle v_3, e_1\rangle e_1 - \langle v_3, e_2\rangle e_2}{||v_3 - \langle v_3, e_1\rangle e_1 - \langle v_3, e_2\rangle e_2||} \\
e_4 &= \frac{v_4 - \langle v_4, e_1\rangle e_1 - \langle v_4, e_2\rangle e_2 - \langle v_4, e_3\rangle e_3}{||v_4 - \langle v_4, e_1\rangle e_1 - \langle v_4, e_2\rangle e_2 - \langle v_4, e_3\rangle e_3||} \\
&\vdots \\
e_k &= \frac{v_k - < v_k, e_1 > e_1 - < v_k, e_2 > e_2 - \cdots - v_k < e_{k-1} > e_{k-1}}{||v_k - < v_k, e_1 > e_1 - < v_k, e_2 > e_2 - \cdots - v_k < e_{k-1} > e_{k-1}||}
\end{aligned}
$$

Runtime

Define n to be the total number of vectors in the set
Define m to be the dimension of the inner product of two vectors in V
Define $(+, \cdot, ||)$ operations to be $O(1)$ time.

The lower bound is O(n)
Assuming that all computations within this loop are $> O(1)$, this mandates a lower bound of $O(n)$ runtime.

We examine the number of computations in the numerator of e_k

e_1 has 0 computations of $O(1)$ and 0 computations of $O(m)$

e_2 has 2 computations of $O(1)$ and 1 computation of $O(m)$

e_3 has 4 computations of $O(1)$ and 2 computations of $O(m)$

e_4 has 6 computations of $O(1)$ and 3 computations of $O(m)$

$\vdots$

e_k has $(n-1) \cdot 2$ computations of $O(1)$ and $(n-1)$ computations of $O(m)$

We examine the number of computations in the denominator of e_k

e_1 has 2 computations of $O(1)$ and 0 computations of $O(m)$

e_2 has 3 computations of $O(1)$ and 1 computation of $O(m)$

e_3 has 5 computations of $O(1)$ and 2 computations of $O(m)$

e_4 has 7 computations of $O(1)$ and 3 computations of $O(m)$

$\vdots$

e_k has $(n-1) \cdot 2$ computations of $O(1)$ and $(n-1)$ computations of $O(m)$

We can notice that the number of $O(1)$ and $O(m)$ computations increases linearly and so can easily be written as a function of n:
Numerator has $(n-1) \cdot 2$ number of $O(1)$ computations and $(n-1)$ $O(m)$ computations
Denominator has $(n-1) \cdot 2 + 1$ number of $O(1)$ computations and $(n-1)$ $O(m)$ computations

If we sum up these runtimes we get

$$(n-1) \cdot 2 + 2 \cdot (n-1) \cdot m + (n-1) \cdot 2 + 1$$

$$= 4 \cdot (n-1) + 2m \cdot (n-1) = (n-1) \cdot (4 + 2m) = 4n - 4 + 2mn - 2m$$

In Big-O we are to discard scalars and floating additions which gives us just

$$O(mn)$$

calculations for every iteration of the loop.

The Gram Schmidt Algorithm runtime is $\mathbf{O(n^2}m)$

Since $O(n) \cdot O(nm) = O(n^2m)$